

NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL

_S

Ps

NP

NP

\$G

\$O

NP

PA

_L

```

NN      NN      MM      MM      LL      SSSSSSSS  HH      HH      000000  WW      WW
NN      NN      MM      MM      LL      SSSSSSSS  HH      HH      000000  WW      WW
NN      NN      MMMM   MMMM   LL      SS          HH      HH      00      00  WW      WW
NN      NN      MMMM   MMMM   LL      SS          HH      HH      00      00  WW      WW
NNNN    NN      MM      MM      LL      SS          HH      HH      00      00  WW      WW
NNNN    NN      MM      MM      LL      SSSSSS      HHHHHHHHHH  00      00  WW      WW
NN  NN  NN      MM      MM      LL      SSSSSS      HHHHHHHHHH  00      00  WW      WW
NN      NN      MM      MM      LL      SS          HH      HH      00      00  WW      WW
NN      NN      MM      MM      LL      SS          HH      HH      00      00  WW      WW
NN      NN      MM      MM      LL      SS          HH      HH      00      00  WW      WW
NN      NN      MM      MM      LL      SSSSSSSS  HH      HH      000000  WW      WW
NN      NN      MM      MM      LLLLLLLLLL  SSSSSSSS  HH      HH      000000  WW      WW
NN      NN      MM      MM      LLLLLLLLLL  SSSSSSSS  HH      HH      000000  WW      WW

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```



```
0001 0 %TITLE 'NML SHOW parameter module'
0002 0 MODULE NML$SHOW (
0003 0     LANGUAGE (BLISS32),
0004 0     ADDRESSING_MODE (EXTERNAL=GENERAL),
0005 0     ADDRESSING_MODE (NONEXTERNAL=GENERAL),
0006 0     IDENT = 'V04-000'
0007 0 ) =
0008 1 BEGIN
0009 1
0010 1 *****
0011 1 *
0012 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0013 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0014 1 *   ALL RIGHTS RESERVED.
0015 1 *
0016 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0017 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0018 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0019 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0020 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0021 1 *   TRANSFERRED.
0022 1 *
0023 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0024 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0025 1 *   CORPORATION.
0026 1 *
0027 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0028 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0029 1 *
0030 1 *****
0031 1
0032 1
0033 1
0034 1 ++
0035 1 FACILITY: DECnet-VAX Network Management Listener
0036 1
0037 1 ABSTRACT:
0038 1
0039 1     These routines return volatile data base information in response to
0040 1     an NCP SHOW command message.
0041 1
0042 1 ENVIRONMENT: VAX/VMS Operating System
0043 1
0044 1 AUTHOR: Distributed Systems Software Engineering
0045 1
0046 1 CREATION DATE: 30-DEC-1979
0047 1
0048 1 MODIFIED BY:
0049 1
0050 1     V03-015 MKP0019      Kathy Perko      4-Mar-1984
0051 1     Fix area numbers when doing SHOW to Phase III nodes.
0052 1
0053 1     V03-014 MKP0018      Kathy Perko      9-Jan-1984
0054 1     Add X25-Access Module entity.
0055 1
0056 1     V03-013 MKP0017      Kathy Perko      9-Nov-1983
0057 1     Fix SHOW KNOWN NODE CIRCUIT <circ id> to simply return a
```



```
58      0058 1      prompt if there aren't any.
59      0059 1
60      0060 1      V03-012 MKP0016      Kathy Perko      31-May-1983
61      0061 1      Fix SHOW single CIRCUIT COUNTERS to return proper data.
62      0062 1
63      0063 1      V03-011 MKP0015      Kathy Perko      6-May-1983
64      0064 1      Fix SHOW CIRCUIT to return circuit info once. Also, fix
65      0065 1      SHOW CIRCUIT to return service adjacency info (SDI database)
66      0066 1      only for NI circuits.
67      0067 1
68      0068 1      V03-010 MKP0014      Kathy Perko      30-April-1983
69      0069 1      Add Service Adjacencies to SHOW CIRCUIT.
70      0070 1
71      0071 1      V03-009 MKP0013      Kathy Perko      25-Jan-1983
72      0072 1      Fix SHOW KNOWN and ACTIVE nodes if there's a circuit qualifier.
73      0073 1
74      0074 1      V03-008 MKP0012      Kathy Perko      14-Nov-1982
75      0075 1      Allow CIRCUIT qualifier on SHOW NODE commands.
76      0076 1
77      0077 1      V03-007 MKP0011      Kathy Perko      12-Nov-1982
78      0078 1      Fix SHOW CIRC nnn COL (which was broken when ADJACENT
79      0079 1      NODE qualifier was added).
80      0080 1
81      0081 1      V03-006 MKP0010      Kathy Perko      29-Oct-1982
82      0082 1      Add area entity.
83      0083 1      Change SHOW CIRCUITS to return the first adjacencies
84      0084 1      information in the same NICE message as the circuit's info.
85      0085 1
86      0086 1      V03-005 MKP0009      Kathy Perko      13-Oct-1982
87      0087 1      Add SHOW ADJACENT NODES CIRCUIT <circuit id> and
88      0088 1      SHOW KNOWN CIRCUITS ADJACENT NODE <node id>.
89      0089 1
90      0090 1      V03-004 MKP0008      Kathy Perko      4-Oct-1982
91      0091 1      Add SHOW ADJACENT NODES and SHOW CIRCUIT(S) ADJACENT NODE(S).
92      0092 1      Add X25-Tracepoints to NML$GET_ENTITY_IDS.
93      0093 1
94      0094 1      V03-003 MKP0007      Kathy Perko      19-Sept-1982
95      0095 1      Redo SHOW KNOW NODES and LOOP NODES to use the multiple
96      0096 1      capabilities of the new QIO interface with NETACP.
97      0097 1
98      0098 1      V03-002 MKP0006      Kathy Perko      1-July-1982
99      0099 1      Add qualifiers to SHOW. Rewrite a bunch of routines in the
100     0100 1      process to take advantage of the enhanced QIO interface.
101     0101 1      Add X29-Server entity.
102     0102 1
103     0103 1      V03-001 MKP0005      Kathy Perko      7-May-1982
104     0104 1      Add double search keys to NETACP QIO interface. Also, combine
105     0105 1      the show active and show known node routines into one.
106     0106 1
107     0107 1      V02-004 MKP0004      Kathy Perko      2-Jan 1982
108     0108 1      Fix SHOW LINKS WITH NODE so that, if the node
109     0109 1      address is greater than 255, the show will work.
110     0110 1
111     0111 1      V02-003 MKP0003      Kathy Perko      21-Oct-1981
112     0112 1      Make NML$GETDATA and NML$PROCESSDATA global
113     0113 1      routines so compatibility module can use them.
114     0114 1
```


NML\$SHOW
V04-000

NML SHOW parameter module

F 4
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[NML.SRC]NMLSHOW.B32;1 Page 3 (1)

:	115	0115	1	:	V02-002	MKP0002	Kathy Perko	8-Sept-1981
:	116	0116	1	:		Fix SHOW EXECUTOR COUNTER		
:	117	0117	1	:				
:	118	0118	1	:	V02-001	MKP0001	Kathy Perko	22-July-1981
:	119	0119	1	:		Add circuit entity and multidrop lines.		
:	120	0120	1	:				
:	121	0121	1	:				


```
123 0122 1 %SBTTL 'Declarations'
124 0123 1
125 0124 1
126 0125 1 : TABLE OF CONTENTS:
127 0126 1
128 0127 1
129 0128 1 FORWARD ROUTINE
130 0129 1 NML$SHOWENTITY,
131 0130 1 NML$SHOWMULTIPLE : NOVALUE,
132 0131 1 NML$PROCESS_MULT_BUFFER: NOVALUE,
133 0132 1 NML$SHOW_CIRCUIT : NOVALUE,
134 0133 1 NML$SHOW_ADJACENCIES,
135 0134 1 NML$SHOW_KNOWN_LOOP : NOVALUE,
136 0135 1 NML$SHOWNODEBYNAME : NOVALUE,
137 0136 1 NML$SHOWEXECUTOR : NOVALUE,
138 0137 1 NML$SHOW_MULTIPLE_NODES: NOVALUE,
139 0138 1 NML$GET_ENTITY_IDS,
140 0139 1 NML$BLD$SHOWBUFS,
141 0140 1 NML$GETDATA,
142 0141 1 NML$PROCESSDATA : NOVALUE,
143 0142 1 NML$GETIDSTRING;
144 0143 1
145 0144 1 : INCLUDE FILES:
146 0145 1
147 0146 1
148 0147 1
149 0148 1 LIBRARY 'LIB$:NMLLIB.L32';
150 0149 1 LIBRARY 'SHRLIB$:NMLIBRY.L32';
151 0150 1 LIBRARY 'SHRLIB$:NET.L32';
152 0151 1 LIBRARY 'SYSS$LIBRARY:STARLET.L32';
153 0152 1
154 0153 1 : OWN STORAGE:
155 0154 1
156 0155 1
157 0156 1
158 0157 1 OWN
159 0158 1 NML$T_LISTBUFFER : VECTOR [NML$K_QIOBFLEN, BYTE];
160 0159 1 BIND
161 0160 1 NML$Q_LISTBFDSC = UPLIT (NML$K_QIOBFLEN, NML$T_LISTBUFFER) : DESCRIPTOR;
162 0161 1
163 0162 1 OWN
164 0163 1 NML$T_P2BUFFER : VECTOR [NML$K_P2BUFLLEN];
165 0164 1 BIND
166 0165 1 NML$Q_P2BFDSC = UPLIT (NML$K_P2BUFLLEN, NML$T_P2BUFFER) : DESCRIPTOR;
167 0166 1
168 0167 1 OWN
169 0168 1 NML$T_ENTBUFFER : VECTOR [32];
170 0169 1 BIND
171 0170 1 NML$Q_ENTBFDSC = UPLIT (32, NML$T_ENTBUFFER) : DESCRIPTOR;
172 0171 1
173 0172 1 OWN
174 0173 1 NML$B_ADJACENCY_FOUND: BYTE;
175 0174 1
176 0175 1 : EXTERNAL REFERENCES:
177 0176 1
178 0177 1
179 0178 1
```


NML\$SHOW
V04-000

NML SHOW parameter module
Declarations

H 4
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[NML.SRC]NMLSHOW.B32;1
Page 5
(2)

```
: 180      0179 1 $NML_EXTDEF;
: 181      0180 1
: 182      0181 1 EXTERNAL
: 183      0182 1      nml$gb_ncp_version,      ! NICE version being spoken
: 184      0183 1      nml$gw_vol_exec_addr : BBLOCK [2];
: 185      0184 1
: 186      0185 1 EXTERNAL LITERAL
: 187      0186 1      CPT$GK_PCNO_DLI;
: 188      0187 1
: 189      0188 1 EXTERNAL ROUTINE
: 190      0189 1      NML$BLD_REPLY,
: 191      0190 1      NML$BLDP2,
: 192      0191 1      NML$ERROR_1,
: 193      0192 1      NML$ERROR_2,
: 194      0193 1      NML$GETEXEADR,
: 195      0194 1      NML$GETINFTABS,
: 196      0195 1      NML$GETNODADR,
: 197      0196 1      NML$GETNODNAM,
: 198      0197 1      NML$NETQIO,
: 199      0198 1      NML$SEND,
: 200      0199 1      NML$SHOWPARLIST;
: 201      0200 1
```

NML
V04

```
0201 1 %SBTTL 'NML$SHOWENTITY Show volatile entity parameters'
0202 1 GLOBAL ROUTINE NML$SHOWENTITY (ENTITY, INF, LEN, ADR) =
0203 1
0204 1 !++
0205 1 FUNCTIONAL DESCRIPTION:
0206 1
0207 1 This routine shows volatile entity parameters.
0208 1
0209 1 FORMAL PARAMETERS:
0210 1
0211 1 ENTITY Entity ID
0212 1 INF Information type code.
0213 1 LEN Length of entity id string.
0214 1 ADR Address of entity id string.
0215 1
0216 1 --
0217 1
0218 2 BEGIN
0219 2
0220 2 LOCAL
0221 2 STATUS,
0222 2 P4_DATA_DSC : DESCRIPTOR, ! QIO data descriptor
0223 2 P4_DATA_PTR, ! Pointer into P4 buffer
0224 2 NICE_MSG_DSC : DESCRIPTOR, ! Output message descriptor
0225 2 NFB_DSC : REF DESCRIPTOR, ! NFB descriptor
0226 2 P2DSC : DESCRIPTOR, ! P2 parameter descriptor
0227 2 TABDES : REF DESCRIPTOR; ! Information table descriptor
0228 2
0229 2
0230 2 !
0231 2 ! Get NFB, table, and P2 buffer.
0232 2
0233 2 NML$GETINFTABS (.ENTITY, .INF, NFB_DSC, TABDES, 0);
0234 2
0235 2 !
0236 2 ! X25 and X29 Server databases have only one entry. So always do a
0237 2 ! wildcard zero of these databases.
0238 2
0239 2 IF .ENTITY EQL NML$C_X25_SERV OR
0240 2 .ENTITY EQL NML$C_X29_SERV OR
0241 2 .ENTITY EQL NML$C_TRACE THEN
0242 2 LEN = -1;
0243 2
0244 2 NML$BLDP2 (.LEN, .ADR, -1, 0, NML$Q_P2BFDSC, P2DSC);
0245 2
0246 2 STATUS = NML$GETDATA (.NFB_DSC, P2DSC, NML$Q_QIOBFDSC, P4_DATA_DSC);
0247 2 IF .STATUS THEN
0248 2 BEGIN
0249 2 P4_DATA_PTR = .P4_DATA_DSC [DSC$A_POINTER];
0250 2 NML$PROCESSDATA (.ENTITY, .TABDES, P4_DATA_DSC, P4_DATA_PTR, NICE_MSG_DSC);
0251 2 END
0252 2 ELSE
0253 2 BEGIN
0254 2 NML$BLD_REPLY (NML$AB_MSGBLOCK, NICE_MSG_DSC [DSC$W_LENGTH]);
0255 2 NICE_MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
0256 2 END;
0257 2
0258 2 NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER], .NICE_MSG_DSC [DSC$W_LENGTH]);
0259 2 RETURN .STATUS;
```


NML\$SHOW
V04-000

; 260

NML SHOW parameter module
NML\$SHOWENTITY Show volatile entity parameters

0258 1 END;

! End of NML\$SHOWENTITY

J 4
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1 Page 7
(3)

.TITLE NML\$SHOW NML SHOW parameter module
.IDENT \V04-000\

.PSECT \$PLITS,NOWRT,NOEXE,2

000004B0	00000	P.AAA:	.LONG	1200
00000000	00004		.ADDRESS	NML\$T_LISTBUFFER
00000068	00008	P.AAB:	.LONG	104
00000000	0000C		.ADDRESS	NML\$T_P2BUFFER
00000020	00010	P.AAC:	.LONG	32
00000000	00014		.ADDRESS	NML\$T_ENTBUFFER

.PSECT \$OWNS,NOEXE,2

00000 NML\$T_LISTBUFFER:
 .BKLB 1200
004B0 NML\$T_P2BUFFER:
 .BKLB 416
00650 NML\$T_ENTBUFFER:
 .BKLB 128
006D0 NML\$B_ADJACENCY_FOUND:
 .BKLB 1

NML\$Q_LISTBFDSC=	P.AAA
NML\$Q_P2BFDSC=	P.AAB
NML\$Q_ENTBFDSC=	P.AAC

.EXTRN NML\$GB_EVTSRCTYP
.EXTRN NML\$GQ_EVTSRCDSC
.EXTRN NML\$GW_EVTCLASS
.EXTRN NML\$GB_EVTMSKTYP
.EXTRN NML\$GQ_EVTMSKDSC
.EXTRN NML\$GW_EVTSNKADR
.EXTRN NML\$GW_ACP_CHAN
.EXTRN NML\$GL_LOGMASK, NML\$GQ_ENTSTRDSC
.EXTRN NML\$AB_QIOBUFFER
.EXTRN NML\$GQ_QIOBFDSC
.EXTRN NML\$AB_EXEBUFFER
.EXTRN NML\$GL_EXEDATPTR
.EXTRN NML\$GQ_EXEDATDSC
.EXTRN NML\$GQ_EXEBFDSC
.EXTRN NML\$AB_RCVBUFFER
.EXTRN NML\$GQ_RCVBFDSC
.EXTRN NML\$AB_SNDBUFFER
.EXTRN NML\$GQ_SNDBFDSC
.EXTRN NML\$GL_RCVDATLEN
.EXTRN NML\$AB_CPTABLE, NML\$AB_MSGBLOCK
.EXTRN NML\$AB_ENTITY_ID
.EXTRN NML\$AB_QUALIFIER_ID
.EXTRN NML\$AB_ENTITYDATA
.EXTRN NML\$AB_NML_NMV, NML\$AB_PRMSEM
.EXTRN NML\$AB_RECBUF, NML\$AL_ENTINFNTAB
.EXTRN NML\$AL_PERMINFTAB
.EXTRN NML\$AW_PRM_DES, NML\$GB_CMD_VER
.EXTRN NML\$GB_ENTITY_CODE

```

                    000C 00000
5E                  24 C2 00002
                    7E D4 00005
                    04 AE 9F 00007
                    0C AE 9F 0000A
                    08 AC DD 0000D
52                  04 AC D0 00010
                    52 DD 00014
00000000G 00      05 FB 00016
11          52 D1 0001D
15          0A 13 00020
13          52 D1 00022
          05 13 00025
          52 D1 00027
          04 12 0002A
          0C AC 01 CE 0002C 1$:
          0C AE 9F 00030 2$:
          00000000' 00 9F 00033
                    7E D4 00039
                    01 CE 0003B
00000000G 7E      0C AC 7D 0003E
7E          06 FB 00042
                    1C AE 9F 00049
          00000000G 00 9F 0004C
                    14 AE 9F 00052
          00000000V 00 AE DD 00055
53          04 FB 00058
1C          50 D0 0005F
08          53 E9 00062
          AE 20 AE D0 00065
                    14 AE 9F 0006A
                    0C AE 9F 0006D
                    24 AE 9F 00070
                    0C AE DD 00073
                    52 DD 00076
```

```
.EXTRN NML$GB_ENTITY_FORMAT
.EXTRN NML$GL_QUALIFIER_PST
.EXTRN NML$GB_QUALIFIER_FORMAT
.EXTRN NML$GB_FUNCTION
.EXTRN NML$GB_INFO, NML$GB_OPTIONS
.EXTRN NML$GL_PRCODE, NML$GL_PRS_FLGS
.EXTRN NML$GL_NML_ENTITY
.EXTRN NML$GQ_NETNAMDSC
.EXTRN NML$GQ_RECBFDSC
.EXTRN NML$GW_PRMDESCNT
.EXTRN NML$GB_NCP_VERSION
.EXTRN NML$GW_VOL_EXEC_ADDR
.EXTRN CPT$GK_PCNO_DLI
.EXTRN NML$BLD_REPCY, NML$BLDP2
.EXTRN NML$ERROR_1, NML$ERKOR_2
.EXTRN NML$GETEXEADR, NML$GETINFTABS
.EXTRN NML$GETNODADR, NML$GETNODNAM
.EXTRN NML$NETQIO, NML$SEND
.EXTRN NML$SHOWPARLIST
```

.PSECT \$CODE\$,NOWRT,2

```
.ENTRY NML$SHOWENTITY, Save R2,R3      : 0202
SUBL2 #36, SP                          :
CLRL -(SP)                             : 0232
PUSHAB TABDES
PUSHAB NFBDS
PUSHL INF
MOVL ENTITY, R2
PUSHL R2
CALLS #5, NML$GETINFTABS
CMPL R2, #17                          : 0237
BEQL 1$
CMPL R2, #21                          : 0238
BEQL 1$
CMPL R2, #19                          : 0239
BNEQ 2$
MNEGL #1, LEN                         : 0240
PUSHAB P2DSC                          : 0242
PUSHAB NML$GQ_P2BFDSC
CLRL -(SP)
MNEGL #1, -(SP)
MOVQ LEN, -(SP)
CALLS #6, NML$BLDP2
PUSHAB P4_DATA_DSC                    : 0244
PUSHAB NML$GQ_QIOBFDSC
PUSHAB P2DSC
PUSHL NFBDS
CALLS #4, NML$GETDATA
MOVL R0, STATUS
BLBC STATUS, 3$                       : 0245
MOVL P4_DATA_DSC+4, P4_DATA_PTR       : 0247
PUSHAB NICE MSG_DSC                   : 0248
PUSHAB P4_DATA_PTR
PUSHAB P4_DATA_DSC
PUSHL TABDES
PUSHL R2
```


NML\$SHOW
V04-000

NML SHOW parameter module

NML\$SHOWENTITY Show volatile entity parameters

L 4
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[NML.SRC]NMLSHOW.B32;1

Page 9
(3)

00000000V	00		05	FB	00078	CALLS	#5, NML\$PROCESSDATA	:	
			18	11	0007F	BRB	4\$:	0245
		14	AE	9F	00081	PUSHAB	NICE_MSG_DSC	:	0252
		00000000G	00	9F	00084	PUSHAB	NML\$AB_MSGBLOCK	:	
00000000G	00		02	FB	0008A	CALLS	#2, NML\$BLD_REPLY	:	
18	AE	00000000G	00	9E	00091	MOVAB	NML\$AB_SNDBUFFER, NICE_MSG_DSC+4	:	0253
	7E	14	AE	3C	00099	MOVZWL	NICE_MSG_DSC, -(SP)	:	0256
		1C	AE	DD	0009D	PUSHL	NICE_MSG_DSC+4	:	
00000000G	00		02	FB	000A0	CALLS	#2, NML\$SEND	:	
	50		53	D0	000A7	MOVL	STATUS, R0	:	0257
			04	000AA	RET			:	0258

; Routine Size: 171 bytes, Routine Base: \$CODE\$ + 0000


```
262 0259 1 %SBTTL 'NML$SHOWMULTIPLE Show multiple entitys parameters'
263 0260 1 GLOBAL ROUTINE NML$SHOWMULTIPLE (ENTITY, INF, FORMAT, ENTITY_ADR,
264 0261 1 QUAL_PST, QUAL_LEN, QUAL_ADR) : NOVALUE =
265 0262 1
266 0263 1 ++
267 0264 1 FUNCTIONAL DESCRIPTION:
268 0265 1
269 0266 1 This routine reads the volatile data base entries for KNOWN or
270 0267 1 ACTIVE entities of the specified type.
271 0268 1
272 0269 1 First the buffers are built which describe the entity type and
273 0270 1 the information required for the SHOW request (STATUS, SUMMARY,
274 0271 1 CHARACTERISTICS, or COUNTERS). These buffers are then given to
275 0272 1 the ACP in a QIO request. The ACP returns the requested information
276 0273 1 for as many entities as will fit in the P4 buffer. The information
277 0274 1 for each entity is formatted into a NICE message and returned to
278 0275 1 NCP. After each circuit is formatted, search the adjacency database
279 0276 1 for all nodes adjacent to that circuit and return a NICE message
280 0277 1 for each node containing it's adjacency information.
281 0278 1
282 0279 1 The QIO is repeated until all entities of the specified type have
283 0280 1 been returned by the ACP.
284 0281 1
285 0282 1
286 0283 1 FORMAL PARAMETERS:
287 0284 1
288 0285 1 ENTITY Entity type code.
289 0286 1 INF Information type code.
290 0287 1 FORMAT NMASC_ENT_KNO => Get KNOWN entities.
291 0288 1 NMASC_ENT_ACT => Get ACTIVE entities.
292 0289 1 NMASC_ENT_ADJ => Get ADJACENT nodes.
293 0290 1 NMASC_ENT_LOO => Get LOOP nodes.
294 0291 1 >0 Length of entity ID (if there is a qualifier on the
295 0292 1 SHOW command, it is essentially a multiple show).
296 0293 1 ENTITY_ADR Used only if there is a qualifier on the command
297 0294 1 because the qualifier makes it essentially a multiple
298 0295 1 SHOW command.
299 0296 1 QUAL_PST Address of qualifier's entry in the Parameter
300 0297 1 Semantic Table (PST).
301 0298 1 QUAL_LEN Length of qualifier ID string.
302 0299 1 QUAL_ADR Address of qualifier ID string.
303 0300 1
304 0301 1 --
305 0302 2 BEGIN
306 0303 2
307 0304 2 LOCAL
308 0305 2 NFB : REF BBLOCK, ! Pointer used to build NFB.
309 0306 2 NFBBUF : BBLOCK [256], ! Buffer in which to build NFB.
310 0307 2 NFBDESC : DESCRIPTOR, ! Pointer to NFB descriptor.
311 0308 2 P2BUF : BBLOCK [NML$K_P2BUFLen], ! P2 buffer
312 0309 2 P2_BUFFER_DESC : DESCRIPTOR, ! Descriptor of empty P2 buffer.
313 0310 2 P2_DESC : DESCRIPTOR, ! Descriptor of P2 contents.
314 0311 2 P4_BUF : BBLOCK [NML$K_QIOBFLen], ! P4 buffer.
315 0312 2 P4_BUFFER_DESC : DESCRIPTOR, ! Descriptor of empty P4 buffer.
316 0313 2 TABDESC : REF DESCRIPTOR, ! Pointer to Information Table desc.
317 0314 2 ENTITY_CNT, ! Count of entities returned by NETACP.
318 0315 2 P4_DATA_DESC : DESCRIPTOR, ! Return P4 buffer descriptor.
```



```
319 0316 2 NICE_MSG_DSC : DESCRIPTOR, ! Output message descriptor
320 0317 2 STATUS;
321 0318 2
322 0319 2
323 0320 2
324 0321 2 ! Get canned NFB and Information Table descriptors for single entity show.
325 0322 2 ! Then modify them to do a plural show.
326 0323 2
327 0324 2 NFB_DSC [DSC$A_POINTER] = NFB_BUF;
328 0325 2 NML$GETINFTABS (.ENTITY, .INF, NFB_DSC, TAB_DSC, 1);
329 0326 2 P2_BUFFER_DSC [DSC$W_LENGTH] = NML$K_P2_BUFLEN;
330 0327 2 P2_BUFFER_DSC [DSC$A_POINTER] = P2_BUF;
331 0328 2 NML$BLD_SHOW_BUFS (.ENTITY, .FORMAT, .ENTITY_ADR,
332 0329 2 NFB_BUF, ! Address of NFB.
333 0330 2 P2_BUFFER_DSC, ! Descriptor of buffer for P2.
334 0331 2 P2_DSC, ! Descriptor for completed P2.
335 0332 2 .QUAL_PST, .QUAL_LEN, .QUAL_ADR); ! Qualifier info
336 0333 2 P4_BUFFER_DSC [DSC$W_LENGTH] = NML$K_P4_BUFLEN;
337 0334 2 P4_BUFFER_DSC [DSC$A_POINTER] = P4_BUF;
338 0335 2 STATUS = T;
339 0336 2 WHILE .STATUS DO
340 0337 3 BEGIN
341 0338 3 STATUS = NML$GETDATA (NFB_DSC, P2_DSC, P4_BUFFER_DSC, P4_DATA_DSC);
342 0339 3 IF .STATUS THEN
343 0340 4 BEGIN
344 0341 4 NML$GL_PRS_FLGS [NML$V_PRS_ENTITY_FOUND] = TRUE;
345 0342 4
346 0343 4 ! The first longword of the P2 buffer contains the number of
347 0344 4 ! entities returned in the P4 buffer. Then call
348 0345 4 ! NML_PROCESS_MULT_BUFFER to return the data in the P4 buffer
349 0346 4 ! to NCP.
350 0347 4
351 0348 4 ENTITY_CNT = (.P2_DSC [DSC$A_POINTER]);
352 0349 4 NML_PROCESS_MULT_BUFFER (.ENTITY, .INF,
353 0350 4 .QUAL_PST, .QUAL_LEN, .QUAL_ADR,
354 0351 4 TAB_DSC, P4_DATA_DSC, .ENTITY_CNT);
355 0352 3 END;
356 0353 2 END;
357 0354 2
358 0355 2 ! Return an error response message to NCP if:
359 0356 2 ! An error other than end-of-file was returned by the ACP.
360 0357 2 ! An end-of-file error was returned by the ACP and
361 0358 2 ! The command had a qualifier and the qualifier wasn't in the volatile
362 0359 2 ! database.
363 0360 2 ! The command was SHOW X-P GROUP yyyy and no such group was found.
364 0361 2
365 0362 2 IF NOT .STATUS THEN
366 0363 3 BEGIN
367 0364 3 IF (.STATUS NEQ NML$STS_CMP) ! If the error wasn't end-of-file
368 0365 3 OR ! or
369 0366 3 ((.STATUS EQL NML$STS_CMP AND ! The error was end-of-file and
370 0367 3 NOT .NML$GL_PRS_FLGS [NML$V_PRS_ENTITY_FOUND]) ! no matches were found in ACPs database
371 0368 3 AND
372 0369 3 ((.NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER]) AND
373 0370 3 (.ENTITY EQL NML$C_PROT_GRP AND ! Entity = X25 group
374 0371 3 .FORMAT GTR 0))) ! Group name specified
375 0372 3
```



```

: 376
: 377
: 378
: 379
: 380
: 381
: 382
: 383
: 384

0373 3 THEN
0374 4 BEGIN
0375 4 NML$BLD REPLY (NML$AB_MSGBLOCK, NICE MSG DSC [DSC$W_LENGTH]);
0376 4 NICE MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
0377 4 NML$SEND(.NICE_MSG_DSC [DSC$A_POINTER],
0378 4 .NICE_MSG_DSC [DSC$W_LENGTH]);
0379 3 END;
0380 2 END;
0381 1 END; ! of NML$SHOWMULTIPLE
```

```

                                001C 00000
                                00 9E 00002
                                CE 9E 00009
FEFC CD FF00 CD 9E 0000E
                                01 DD 00015
                                AE 9F 00017
                                CD 9F 0001A
                                04 AC 7D 0001E
00000000G 00 05 FB 00022
FE88 CD 68 8F 9B 00029
FE8C CD FE90 CD 9E 0002F
                                18 AC 7D 00036
                                14 AC DD 0003A
FE80 CD 9F 0003D
FE88 CD 9F 00041
FE00 CD 9F 00045
                                7E 0C AC 7D 00049
                                04 AC DD 0004D
00000000V 00 09 FB 00050
14 AE 04B0 8F B0 00057
18 AE 1C AE 9E 0005D
52 01 D0 00062
3F 52 E9 00065 1$:
                                0C AE 9F 00068
                                18 AE 9F 0006B
FE80 CD 9F 0006E
FEF8 CD 9F 00072
                                04 FB 00076
00000000V 00 50 D0 0007D
52 52 E9 00080
64 08 88 00083
53 FE84 DD D0 00086
                                53 DD 0008B
                                10 AE 9F 0008D
                                08 AE 9F 00090
                                7E 18 AC 7D 00093
                                14 AC DD 00097
                                7E 04 AC 7D 0009A
00000000V 00 08 FB 0009E
FFFFFFFF0 8F BE 11 000A5
                                52 D1 000A7 2$:
                                13 12 000AE
35 64 03 E0 000B0

.ENTRY NML$SHOWMULTIPLE, Save R2,R3,R4
MOVAB NML$GL_PRS_FLGS, R4
MOVAB -1612(SP), -SP
MOVAB NFBBUF, NFB DSC+4
PUSHL #1
PUSHAB TABDSC
PUSHAB NFB DSC
MOVQ ENTITY, -(SP)
CALLS #5, NML$GETINFTABS
MOVZBW #104, P2_BUFFER_DSC
MOVAB P2BUF, P2_BUFFER_DSC+4
MOVQ QUAL_LEN, -(SP)
PUSHL QUAL_PST
PUSHAB P2_DSC
PUSHAB P2_BUFFER_DSC
PUSHAB NFBBUF
MOVQ FORMAT, -(SP)
PUSHL ENTITY
CALLS #9, NML$BLDSHOWBUFS
MOVW #1200, P4_BUFFER_DSC
MOVAB P4_BUF, P4_BUFFER_DSC+4
MOVL #1, STATUS
BLBC STATUS, 2$
PUSHAB P4_DATA_DSC
PUSHAB P4_BUFFER_DSC
PUSHAB P2_DSC
PUSHAB NFB DSC
CALLS #4, NML$GETDATA
MOVL R0, STATUS
BLBC STATUS, 2$
BISB2 #8, NML$GL_PRS_FLGS
MOVL @P2_DSC+4, ENTITY_CNT
PUSHL ENTITY_CNT
PUSHAB P4_DATA_DSC
PUSHAB TABDSC
MOVQ QUAL_LEN, -(SP)
PUSHL QUAL_PST
MOVQ ENTITY, -(SP)
CALLS #8, NML_PROCESS_MULT_BUFFER
BRB 1$
CMPL STATUS, #-16
BNEQ 3$
BBS #3, NML$GL_PRS_FLGS, 4$

: 0260
: 0324
: 0325
: 0326
: 0327
: 0332
: 0328
: 0333
: 0334
: 0335
: 0336
: 0338
: 0339
: 0341
: 0348
: 0351
: 0349
: 0350
: 0349
: 0336
: 0364
: 0367
```


NML\$SHOW
V04-000

NML SHOW parameter module
NML\$SHOWMULTIPLE Show multiple entitys paramet

C 5
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1

Page 13
(4)

31	64	02	E1	000B4	BBC	#2, NML\$GL_PRS_FLGS, 4\$: 0370	
	10	04	AC	D1	000B8	CMPL	ENTITY, #16	: 0371
			2B	12	000BC	BNEQ	4\$: 0372
		0C	AC	D5	000BE	TSTL	FORMAT	: 0372
			26	15	000C1	BLEQ	4\$: 0375
		04	AE	9F	000C3	PUSHAB	NICE MSG DSC	: 0375
			00	9F	000C6	PUSHAB	NML\$AB MSGBLOCK	: 0376
00000000G	00		02	FB	000CC	CALLS	#2, NML\$BLD REPLY	: 0376
08	AE	00000000G	00	9E	000D3	MOVAB	NML\$AB SNDBUFFER, NICE_MSG_DSC+4	: 0378
	7E		04	AE	3C	MOVZWL	NICE_MSG_DSC, -(SP)	: 0377
		0C	AE	DD	000DF	PUSHL	NICE_MSG_DSC+4	: 0377
00000000G	00		02	FB	000E2	CALLS	#2, NML\$SEND	: 0381
			04	000E9	4\$:	RET		: 0381

; Routine Size: 234 bytes, Routine Base: \$CODE\$ + 00AB


```
386 0382 1 %SBTTL 'NML_PROCESS_MULT_BUFFER Show multiple entitys parameters'
387 0383 1 ROUTINE NML_PROCESS_MULT_BUFFER (ENTITY, INF,
388 0384 1 QUAL_PST, QUAL_LEN, QUAL_ADR,
389 0385 1 TABDSC, P4_DATA_DSC, ENTITIES_IN_P4) : NOVALUE =
390 0386 1
391 0387 1 !++
392 0388 1 FUNCTIONAL DESCRIPTION:
393 0389 1 This routine is called only by NML$SHOWMULTIPLE after it has
394 0390 1 a P4 buffer with the information for a number of entities to
395 0391 1 be returned to NCP. For each entity in the P4 buffer, the
396 0392 1 routine builds a NICE message and sends it back to NCP.
397 0393 1
398 0394 1 FORMAL PARAMETERS:
399 0395 1
400 0396 1 ENTITY Entity ID
401 0397 1 INF Information type code.
402 0398 1 QUAL_PST Address of qualifier's entry in the Parameter
403 0399 1 Semantic Table (PST).
404 0400 1 QUAL_LEN Length of qualifier ID string.
405 0401 1 QUAL_ADR Address of qualifier ID string.
406 0402 1 TABDSC Information table descriptor
407 0403 1 P4_DATA_DSC Descriptor of data in P4 buffer.
408 0404 1 ENTITIES_IN_P4 Number of entities for which there is information
409 0405 1 in the P4 buffer.
410 0406 1 !--
411 0407 1
412 0408 2 BEGIN
413 0409 2
414 0410 2 MAP
415 0411 2 P4_DATA_DSC: REF DESCRIPTOR;
416 0412 2
417 0413 2 !
418 0414 2 ! NFB to show an entry in NETACPs adjacency database.
419 0415 2 !
420 P 0416 2 $NFB DSC (NML$Q_ADJ NFB, SHOW,, AJI
421 P 0417 2 ,ADD, ! Search key 1 = node address
422 P 0418 2 ,CIR, ! Search key 2 = circuit name
423 0419 2 );
424 0420 2
425 0421 2 LOCAL
426 0422 2 NICE_MSG_DSC: DESCRIPTOR, ! NICE response message descriptor.
427 0423 2 P4_DATA_PTR, ! Pointer to data in P4 buffer.
428 0424 2 ENTITY_LEN,
429 0425 2 ENTITY_ADDR,
430 0426 2 STATUS,
431 0427 2 CIRCUIT_TYPE,
432 0428 2
433 0429 2 ! Following are fields used for issuing secondary QIOs to adjacency
434 0430 2 ! database. Used for SHOW ADJACENT NODES CIRCUIT <circuit id>.
435 0431 2 !
436 0432 2 ADJ_P2_BUF: BBLOCK [NML$K_P2BUFLN],
437 0433 2 ADJ_P2_BUF_DSC: DESCRIPTOR, ! Descriptor for empty P2 buffer.
438 0434 2 ADJ_P2_DSC: DESCRIPTOR; ! P2 buffer descriptor
439 0435 2
440 0436 2 P4_DATA_PTR = .P4_DATA_DSC [DSC$A_POINTER];
441 0437 2 ADJ_P2_BUF_DSC [DSC$W_LENGTH] = NML$K_P2BUFLN;
442 0438 2 ADJ_P2_BUF_DSC [DSC$A_POINTER] = ADJ_P2_BUF;
```



```
443 0439 2 WHILE (ENTITIES_IN_P4 = .ENTITIES_IN_P4 - 1) GEQ 0 DO
444 0440 2
445 0441 2     Format the entity's data into NICE response
446 0442 2     message.
447 0443 2
448 0444 2 BEGIN
449 0445 2 STATUS = TRUE;
450 0446 2 SELECTU .ENTITY OF
451 0447 2 SET
452 0448 2
453 0449 2     Save the circuit type for the call to show the service adjacencies.
454 0450 2     Save the circuit ID for the call to show the adjacencies.
455 0451 2
456 0452 2 [NML$C_CIRCUIT]:
457 0453 2 BEGIN
458 0454 2     CIRCUIT_TYPE = ..P4_DATA_PTR;
459 0455 2     P4_DATA_PTR = .P4_DATA_PTR + 4;
460 0456 2     ENTITY_LEN = .(.P4_DATA_PTR) < 0,16>;
461 0457 2     ENTITY_ADDR = .P4_DATA_PTR + 2;
462 0458 2 END;
463 0459 2
464 0460 2     The NICE command is SHOW ADJACENT NODES [CIRCUIT <circuit id>].
465 0461 2
466 0462 2 [NML$C_ADJACENT_NODE]:
467 0463 2 BEGIN
468 0464 2
469 0465 2     If the NICE command is qualified (I.E. SHOW ADJACENT NODES
470 0466 2     CIRCUIT <circuit id>) don't return the node's information
471 0467 2     unless it's in the adjacency database for the specified circuit.
472 0468 2
473 0469 2 IF .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] THEN
474 0470 2 BEGIN
475 0471 2     STATUS = FALSE;
476 0472 2     ENTITY_LEN = 0;
477 0473 2     ENTITY_ADDR = ..P4_DATA_PTR;
478 0474 2     NML$BLDP2 (.ENTITY_LEN, .ENTITY_ADDR,      ! Search 1 = node address
479 0475 2     .QUAL_LEN, .QUAL_ADR,                    ! Search 2 = circuit name
480 0476 2     ADJ_P2_BUF_DSC,                          ! P2 buffer descriptor
481 0477 2     ADJ_P2_DSC);                             ! Return P2 buffer desc.
482 0478 2     STATUS = NML$GETDATA (NML$Q_ADJ_NFB, ADJ_P2_DSC,
483 0479 2     0, 0);
484 0480 2 END;
485 0481 2 END;
486 0482 2
487 0483 2 [ALWAYS]:
488 0484 2
489 0485 2     Build the NICE response message and send it to NCP.
490 0486 2     Status is false only if I am processing a
491 0487 2     SHOW ADJACENT NODES CIRCUIT <circuit id> and the
492 0488 2     node in the P4 buffer is not adjacent on the specified
493 0489 2     circuit.
494 0490 2
495 0491 2 BEGIN
496 0492 2     NML$PROCESSDATA (.ENTITY,
497 0493 2     ..TABDSC,
498 0494 2     .P4_DATA_DSC,
499 0495 2     P4_DATA_PTR,
```



```

      NICE_MSG_DSC);
    IF .STATUS THEN
      BEGIN
        Don't send the NICE message here for circuits. The
        adjacency information for the first adjacency must
        still be added to the message.
      IF .ENTITY NEQ NML$C_CIRCUIT THEN
        NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
                  .NICE_MSG_DSC [DSC$W_LENGTH]);
      END;
    END;
[NML$C_CIRCUIT]:
  For circuits, the first NICE message returned for each circuit
  contains the circuit's information from the NETACPs CRI (circuit)
  database plus the first adjacency information from NETACP's
  AJI (adjacency) or SDI (service adjacency) database. Then the
  subsequent adjacencies are returned one to a NICE message
  containing only the circuit ID and the adjacency information.
  BEGIN
    IF .INF NEQ NML$C_COUNTERS THEN
      BEGIN
        NML$B_ADJACENCY_FOUND = 0;
        STATUS = NML_SHOW_ADJACENCIES (NML$C_CIRCUIT_ADJACENT,
                                         .INF, .ENTITY_LEN, .ENTITY_ADDR,
                                         .QUAL_PST, .QUAL_LEN, .QUAL_ADR,
                                         NICE_MSG_DSC);

        The service adjacency database contains no node information
        (hence no need to look if there's an adjacent node qualifier
        on the command) and applies only to NI circuits.

        IF (NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER]) AND
           .CIRCUIT_TYPE EQC NML$C_CIRTY_NI THEN
          STATUS = NML_SHOW_ADJACENCIES (NML$C_CIRCUIT_ADJ_SRV,
                                         .INF, .ENTITY_LEN, .ENTITY_ADDR,
                                         .QUAL_PST, .QUAL_LEN, .QUAL_ADR,
                                         NICE_MSG_DSC);

          If there is no adjacency information for the circuit in either
          adjacency database and the NICE command isn't qualified by an
          ADJACENT NODE (in which case the lack of adjacency information
          means there's nothing to return), return just the circuit information

        IF .NML$B_ADJACENCY_FOUND EQ 0 AND
           (NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER]) AND
           .STATUS EQ NML$STS_CMP THEN
          NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
                    .NICE_MSG_DSC [DSC$W_LENGTH]);
        END
      ELSE
        NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
                  .NICE_MSG_DSC [DSC$W_LENGTH]);
      END
    END
  END
```


NML\$SHOW
V04-000

NML SHOW parameter module

NML_PROCESS_MULT_BUFFER Show multiple entitys

G 5
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NMLSHOW.B32;1

Page 17
(5)

```
: 557      0553 3      END;
: 558      0554 3
: 559      0555 3      TES;
: 560      0556 2      END;
: 561      0557 1 END;      ! of NML_PROCESS_MULT_BUFFER
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

```
00000014 00018 P.AAD: .LONG 20
00000000 0001C .ADDRESS U.1
```

.PSECT \$OWNS\$,NOEXE,2

```
                006D1 .BLKB 3
                22 006D4 ; NFB
                   U.1: .BYTE 34
                   00 006D5 .BYTE 0
                   13 006D6 .BYTE 19
                   00 006D7 .BYTE 0
13010010 006D8 .LONG 318832656
13020042 006DC .LONG 318898242
                   00 006E0 .BYTE 0
                   00 006E1 .BYTE 0
                   0000 006E2 .WORD 0
00000000 006E4 .LONG 0
```

U.2=

P.AAD

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 NML_PROCESS_MULT_BUFFER:

```
5B 00000000V 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0383
5A 00000000V 00 9E 00009 MOVAB NML_SHOW_ADJACENCIES, R11
59 00000000G 00 9E 00010 MOVAB NML$B_ADJACENCY_FOUND, R10
58 00000000G 00 9E 00017 MOVAB NML$SEND, R9
5E 80 AE 9E 0001E MOVAB NML$GL_PRS_FLGS, R8
53 1C AC D0 00022 MOVAB -128(SP), SP
OC AE 04 A3 DD 00026 MOVL P4_DATA_DSC, R3 : 0436
10 AE 68 8F 9B 00029 PUSHL 4(R3)
AE 14 AE 9E 0002E MOVZBW #104, ADJ_P2_BUF_DSC : 0437
20 AC D7 00033 1$: MOVAB ADJ_P2_BUF, ADJ_P2_BUF_DSC+4 : 0438
01 18 00036 DECL ENTITIES_IN_P4 : 0439
04 00038 BGEQ 2$
56 01 D0 00039 2$: RET
52 04 AC D0 0003C MOVL #1, STATUS : 0445
09 52 D1 00040 MOVL ENTITY, R2 : 0446
OF 12 00043 CMPL R2, #9 : 0452
57 00 BE D0 00045 BNEQ 3$
6E 04 C0 00049 MOVL @P4_DATA_PTR, CIRCUIT_TYPE : 0454
55 00 BE 3C 0004C ADDL2 #4, P4_DATA_PTR : 0455
6E 02 C1 00050 MOVZWL @P4_DATA_PTR, ENTITY_LEN : 0456
06 52 D1 00054 3$: ADDL3 #2, P4_DATA_PTR, ENTITY_ADDR : 0457
36 12 00057 CMPL R2, #6 : 0462
BNEQ 4$
```

54

32	68	02	E1	00059	BBC	#2, NML\$GL_PRS_FLGS, 4\$	0469	
		55	7C	0005D	CLRQ	ENTITY_LEN	0472	
	54	00	BE	D0	0005F	MOVL	@P4_DATA_PTR, ENTITY_ADDR	0473
		04	AE	9F	00063	PUSHAB	ADJ_P2_DSC	0474
		10	AE	9F	00066	PUSHAB	ADJ_P2_BUF_DSC	
		14	BC	DD	00069	PUSHL	@QUAL_ADR	0475
		10	AC	DD	0006C	PUSHL	QUAL_LEN	
		54	DD	0006F	PUSHL	ENTITY_ADDR	0474	
		55	DD	00071	PUSHL	ENTITY_LEN		
00000000G	00	06	FB	00073	CALLS	#6, NML\$BLDP2		
		7E	7C	0007A	CLRQ	-(SP)	0478	
		0C	AE	9F	0007C	PUSHAB	ADJ_P2_DSC	
	00000000V	00	9F	0007F	PUSHAB	U.2		
	56	04	FB	00085	CALLS	#4, NML\$GETDATA		
		50	D0	0008C	MOVL	R0, STATUS		
		7C	AE	9F	0008F	PUSHAB	NICE_MSG_DSC	0492
		04	AE	9F	00092	PUSHAB	P4_DATA_PTR	
		53	DD	00095	PUSHL	R3	0494	
		18	BC	DD	00097	PUSHL	@TABDSC	0493
		04	AC	DD	0009A	PUSHL	ENTITY	0492
00000000V	00	05	FB	0009D	CALLS	#5, NML\$PROCESSDATA		
	10	56	E9	000A4	BLBC	STATUS, 5\$	0497	
	09	04	AC	D1	000A7	CMPL	ENTITY, #9	0504
		0A	13	000AB	BEQL	5\$		
	7E	7C	AE	3C	000AD	MOVZWL	NICE_MSG_DSC, -(SP)	0506
		FC	AD	DD	000B1	PUSHL	NICE_MSG_DSC+4	0505
	69	02	FB	000B4	CALLS	#2, NML\$SEND		
	09	52	D1	000B7	CMPL	R2, #9	0510	
		5E	12	000BA	BNEQ	8\$		
	03	08	AC	D1	000BC	CMPL	INF, #3	0520
		4E	13	000C0	BEQL	7\$		
		6A	94	000C2	CLRB	NML\$B_ADJACENCY_FOUND	0522	
		7C	AE	9F	000C4	PUSHAB	NICE_MSG_DSC	0523
	7E	10	AC	7D	000C7	MOVQ	QUAL_LEN, -(SP)	0525
		0C	AC	DD	000CB	PUSHL	QUAL_PST	
		54	DD	000CE	PUSHL	ENTITY_ADDR	0524	
		55	DD	000D0	PUSHL	ENTITY_LEN		
		08	AC	DD	000D2	PUSHL	INF	
		0A	DD	000D5	PUSHL	#10	0523	
	68	08	FB	000D7	CALLS	#8, NML_SHOW_ADJACENCIES		
1E	56	50	D0	000DA	MOVL	R0, STATUS		
	68	02	E0	000DD	BBS	#2, NML\$GL_PRS_FLGS, 6\$	0532	
	06	57	D1	000E1	CMPL	CIRCUIT_TYPE, #6	0533	
		19	12	000E4	BNEQ	6\$		
		7C	AE	9F	000E6	PUSHAB	NICE_MSG_DSC	0534
	7E	10	AC	7D	000E9	MOVQ	QUAL_LEN, -(SP)	0536
		0C	AC	DD	000ED	PUSHL	QUAL_PST	
		54	DD	000F0	PUSHL	ENTITY_ADDR	0535	
		55	DD	000F2	PUSHL	ENTITY_LEN		
		08	AC	DD	000F4	PUSHL	INF	
		0B	DD	000F7	PUSHL	#11	0534	
	68	08	FB	000F9	CALLS	#8, NML_SHOW_ADJACENCIES		
	56	50	D0	000FC	MOVL	R0, STATUS		
		6A	95	000FF	TSTB	NML\$B_ADJACENCY_FOUND	0544	
		17	12	00101	BNEQ	8\$		
13		02	E0	00103	BBS	#2, NML\$GL_PRS_FLGS, 8\$	0545	
FFFFFFF0	68	56	D1	00107	CMPL	STATUS, #-T6	0546	

NML\$SHOW
VO4-000

NML SHOW parameter module
NML_PROCESS_MULT_BUFFER Show multiple entitys

I 5
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VM\$MASTER:[NML.SRC]NML\$SHOW.B32;1

Page 19
(5)

7E 7C 0A 12 0010E
FC AE 3C 00110 7\$:
AD DD 00114
02 FB 00117
69 FF16 31 0011A 8\$:
04 0011D

BNEQ 8\$
MOVZWL NICE-MSG-DSC, -(SP)
PUSHL NICE-MSG-DSC+4
CALLS #2, NML\$SEND
BRW 1\$
RET

: 0552
: 0551
: 0439
: 0557

; Routine Size: 286 bytes, Routine Base: \$CODE\$ + 0195

NML
VO4


```
563 0558 1 %SBTTL 'NML$SHOW CIRCUIT Show volatile circuit parameters'
564 0559 1 GLOBAL ROUTINE NML$SHOW_CIRCUIT (ENTITY, INF, FORMAT, ENTITY_ADR,
565 0560 1 QUAL_PST, QUAL_LEN, QUAL_ADR) : NOVALUE =
566 0561 1
567 0562 1 !++
568 0563 1 FUNCTIONAL DESCRIPTION:
569 0564 1 This routine shows volatile circuit parameters.
570 0565 1
571 0566 1 FORMAL PARAMETERS:
572 0567 1
573 0568 1 ENTITY Entity ID
574 0569 1 INF Information type code.
575 0570 1 FORMAT Entity format or length of entity id string.
576 0571 1 ENTITY_ADR Address of entity id string.
577 0572 1 QUAL_PST Address of qualifier's entry in the Parameter
578 0573 1 Semantic Table (PST).
579 0574 1 QUAL_LEN Length of qualifier ID string.
580 0575 1 QUAL_ADR Address of qualifier ID string.
581 0576 1
582 0577 1 !--
583 0578 1
584 0579 2 BEGIN
585 0580 2
586 0581 2 First, return the information in the circuit database.
587 0582 2
588 0583 2 LOCAL
589 0584 2 STATUS,
590 0585 2 P4_DATA_DSC : DESCRIPTOR, ! QIO data descriptor
591 0586 2 P4_DATA_PTR, ! Pointer into P4 buffer
592 0587 2 NICE_MSG_DSC : DESCRIPTOR, ! Output message descriptor
593 0588 2 NFB_DSC : REF DESCRIPTOR, ! NFB descriptor
594 0589 2 P2_DSC : DESCRIPTOR, ! P2 parameter descriptor
595 0590 2 TABDES : REF DESCRIPTOR, ! Information table descriptor
596 0591 2 CIRCUIT_TYPE;
597 0592 2
598 0593 2 !
599 0594 2 Get NFB, table, and P2 buffer.
600 0595 2
601 0596 2 NML$GETINFABS (.ENTITY, .INF, NFB_DSC, TABDES, 0);
602 0597 2
603 0598 2 NML$BLDP2 (.FORMAT, .ENTITY_ADR, -1, 0, NML$Q_P2BFDSC, P2_DSC);
604 0599 2
605 0600 2 STATUS = NML$GETDATA (.NFB_DSC, P2_DSC, NML$GQ_QIOBFDSC, P4_DATA_DSC);
606 0601 2 IF .STATUS THEN
607 0602 2 BEGIN
608 0603 2 P4_DATA_PTR = .P4_DATA_DSC [DSC$A_POINTER];
609 0604 2 CIRCUIT_TYPE = ..P4_DATA_PTR;
610 0605 2 P4_DATA_PTR = .P4_DATA_PTR + 4;
611 0606 2 NML$PROCESSDATA (.ENTITY, .TABDES, P4_DATA_DSC, P4_DATA_PTR, NICE_MSG_DSC);
612 0607 2
613 0608 2 ! Now, return the information from NETACPs adjacency database (AJI) and
614 0609 2 service adjacency database (SDI). If the SHOW command specifies a node,
615 0610 2 it is specified in the qualifier information, so only that adjacent
616 0611 2 node's information will be returned.
617 0612 2
618 0613 2 IF .INF NEQ NML$C_COUNTERS THEN
619 0614 2 BEGIN
```

```
620 0615 4 NML$B_ADJACENCY_FOUND = 0;
621 0616 4 STATUS = NML_SHOW_ADJACENCIES (NML$C_CIRCUIT_ADJACENT,
622 0617 4 .INF, .FORMAT, .ENTITY_ADR,
623 0618 4 .QUAL_PST, .QUAL_LEN, .QUAL_ADR,
624 0619 4 NICE_MSG_DSC);
625 0620 4
626 0621 4 | The service adjacency database contains no node information
627 0622 4 | (hence no need to look if there's an adjacent node qualifier
628 0623 4 | on the command) and applies only to NI circuits.
629 0624 4
630 0625 4 IF (NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER]) AND
631 0626 4 .CIRCUIT_TYPE EQC NML$C_CIRTY_NI THEN
632 0627 4 STATUS = NML_SHOW_ADJACENCIES (NML$C_CIRCUIT_ADJ_SRV,
633 0628 4 .INF, .FORMAT, .ENTITY_ADR,
634 0629 4 .QUAL_PST, .QUAL_LEN, .QUAL_ADR,
635 0630 4 NICE_MSG_DSC);
636 0631 4
637 0632 4 | If there is no adjacency information for the circuit in either
638 0633 4 | adjacency database and the NICE command isn't qualified by an
639 0634 4 | ADJACENT NODE (in which case the lack of adjacency information
640 0635 4 | means there's nothing to return), return just the circuit information
641 0636 4
642 0637 4 IF .NML$B_ADJACENCY_FOUND EQC 0 AND
643 0638 4 (NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER]) AND
644 0639 4 .STATUS EQC NML$STS_CMP THEN
645 0640 4 NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
646 0641 4 .NICE_MSG_DSC [DSC$W_LENGTH]);
647 0642 4 END
648 0643 3 ELSE
649 0644 3 NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER], .NICE_MSG_DSC [DSC$W_LENGTH]);
650 0645 3 END
651 0646 2 ELSE
652 0647 2 BEGIN
653 0648 2 NML$BLD_REPLY (NML$AB_MSGBLOCK, NICE_MSG_DSC [DSC$W_LENGTH]);
654 0649 2 NICE_MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
655 0650 2 END;
656 0651 2
657 0652 2 RETURN .STATUS;
658 0653 1 END;

! End of NML$SHOWCIRCUIT
```

```
007C 00000
56 00000000G 00 9E 00002
55 00000000V 00 9E 00009
54 00000000' 00 9E 00010
5E 24 C2 00017
7E D4 0001A
04 AE 9F 0001C
0C AE 9F 0001F
7E 04 AC 7D 00022
00000000G 00 05 FB 00026
0C AE 9F 0002D
00000000' 00 9F 00030
7E D4 00036

.ENTRY NML$SHOW_CIRCUIT, Save R2,R3,R4,R5,R6 ; 0559
MOVAB NML$GL_PRS_FLGS, R6
MOVAB NML_SHOW_ADJACENCIES, R5
MOVAB NML$B_ADJACENCY_FOUND, R4
SUBL2 #36, SP
CLRL -(SP)
PUSHAB TABDES
PUSHAB NFBDS
MOVQ ENTITY, -(SP)
CALLS #5, NML$GETINFTABS
PUSHAB P2DSC
PUSHAB NML$Q_P2BFDSC
CLRL -(SP) ; 0596
; 0598
```


00000000G	7E 7E 00	0C	01 AC 06 AE 00 14 AE 10	CE 00038 7D 0003B FB 0003F 9F 00046 9F 00049 9F 0004F DD 00052 FB 00055 DO 0005C E8 0005F 31 00062 DO 00065 DO 0006A CO 0006E 9F 00072 9F 00075 9F 00078 DD 0007B DD 0007E 05 FB 00081 AC D1 00088 4A 13 0008C 94 0008E 9F 00090 AC 7D 00093 AC 7D 00097 AC 7D 0009B DA DD 0009F 08 FB 000A1 50 DO 000A4 02 E0 000A7 52 D1 000AB 17 12 000AE AE 9F 000B0 AC 7D 000B3 AC 7D 000B7 AC 7D 000BB 0B DD 000BF 08 FB 000C1 50 DO 000C4 64 95 000C7 34 12 000C9 02 E0 000CB 53 D1 000CF 27 12 000D6 AE 3C 000D8 AE DD 000DC 02 FB 000DF 04 000E6 AE 9F 000E7 00 9F 000EA 02 FB 000F0 9E 000F7 04 000FF	1\$: 1\$: 2\$: 3\$: 4\$: 5\$:	MNEGL #1, -(SP) MOVQ FORMAT, -(SP) CALLS #6, NML\$BLDP2 PUSHAB P4 DATA_DSC PUSHAB NML\$GQ_010BFDSC PUSHAB P2DSC PUSHL NFB DSC CALLS #4, NML\$GETDATA MOVL R0, STATUS BLBS STATUS, 1\$ BRW 4\$ MOVL P4 DATA_DSC+4, P4 DATA_PTR MOVL @P4 DATA_PTR, CIRCUIT_TYPE ADDL2 #4, P4 DATA_PTR PUSHAB NICE_MSG_DSC PUSHAB P4 DATA_PTR PUSHAB P4 DATA_DSC PUSHL TABDES PUSHL ENTITY CALLS #5, NML\$PROCESSDATA CMPL INF, #3 BEQL 3\$ CLRB NML\$B_ADJACENCY_FOUND PUSHAB NICE_MSG_DSC MOVQ QUAL_LEN, -(SP) MOVQ ENTITY_ADR, -(SP) MOVQ INF, -(SP) PUSHL #10 CALLS #8, NML_SHOW_ADJACENCIES MOVL R0, STATUS BBS #2, NML\$GL_PRS_FLGS, 2\$ CMPL CIRCUIT_TYPE, #6 BNEQ 2\$ PUSHAB NICE_MSG_DSC MOVQ QUAL_LEN, -(SP) MOVQ ENTITY_ADR, -(SP) MOVQ INF, -(SP) PUSHL #11 CALLS #8, NML_SHOW_ADJACENCIES MOVL R0, STATUS TSTB NML\$B_ADJACENCY_FOUND BNEQ 5\$ BBS #2, NML\$GL_PRS_FLGS, 5\$ CMPL STATUS, #-T6 BNEQ 5\$ MOVZWL NICE_MSG_DSC, -(SP) PUSHL NICE_MSG_DSC+4 CALLS #2, NML\$SEND RET PUSHAB NICE_MSG_DSC PUSHAB NML\$AB_MSGBLOCK CALLS #2, NML\$BLD_REPLY MOVAB NML\$AB_SNDBUFFER, NICE_MSG_DSC+4 RET	0600 0601 0603 0604 0605 0606 0613 0615 0616 0618 0617 0616 0625 0626 0627 0629 0628 0627 0637 0638 0639 0644 0601 0648 0649 0653
-----------	----------------	----	--	--	--	---	--

; Routine Size: 256 bytes, Routine Base: \$CODE\$ + 02B3

NML\$SHOW
V04-C00

NML SHOW parameter module
NML\$SHOW_CIRCUIT

Show volatile circuit parame

M 5
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1
Page 23
(6)

NML
V04


```

: 660 0654 1 %SBTTL 'NML_SHOW_ADJACENCIES Show circuit node adjacencies'
: 661 0655 1 ROUTINE NML_SHOW_ADJACENCIES (ENTITY, INF, ENTITY_LEN, ENTITY_ADDR,
: 662 0656 1 QUAL_PST, QUAL_LEN, QUAL_ADR,
: 663 0657 1 NICE_MSG_DSC) =
: 664 0658 1
: 665 0659 1 !++
: 666 0660 1 FUNCTIONAL DESCRIPTION:
: 667 0661 1 This routine is called for SHOW CIRCUIT commands. It is called after
: 668 0662 1 the circuit's information has been retrieved from NETACP's CRI database
: 669 0663 1 and formatted into a NICE message. This routine gets buffers of
: 670 0664 1 adjacency information for the circuit from NETACPs AJI database.
: 671 0665 1 The first adjacency is added to the NICE message containing the circuit's
: 672 0666 1 info from the CRI database. The others are all returned in individual
: 673 0667 1 NICE messages.
: 674 0668 1
: 675 0669 1
: 676 0670 1 FORMAL PARAMETERS:
: 677 0671 1
: 678 0672 1 ENTITY Entity ID
: 679 0673 1 INF Information type code.
: 680 0674 1 ENTITY_LEN Length of circuit ID
: 681 0675 1 ENTITY_ADDR Pointer to circuit ID string.
: 682 0676 1 QUAL_PST Address of qualifier's entry in the Parameter
: 683 0677 1 Semantic Table (PST).
: 684 0678 1 QUAL_LEN Length of qualifier ID string.
: 685 0679 1 QUAL_ADR Address of qualifier ID string.
: 686 0680 1 NICE_MSG_DSC Address of descriptor of NICE message which contains
: 687 0681 1 circuit info. Add the first adjacency info to this
: 688 0682 1 message.
: 689 0683 1 !--
: 690 0684 1
: 691 0685 2 BEGIN
: 692 0686 2
: 693 0687 2 MAP
: 694 0688 2 NICE_MSG_DSC: REF DESCRIPTOR;
: 695 0689 2
: 696 0690 2 LOCAL
: 697 0691 2 P4_DATA_PTR, ! Pointer to data in P4 buffer.
: 698 0692 2 STATUS,
: 699 0693 2 ADJ_NFB_BUF: BBLOCK [256], ! Buffer for adjacency data base NFB.
: 700 0694 2 ADJ_NFB_DSC: DESCRIPTOR, ! NFB descriptor
: 701 0695 2 ADJ_TABDSC: REF DESCRIPTOR, ! Information table descriptor
: 702 0696 2 ADJ_P2_BUF: BBLOCK [NML$K_P2BUFLen],
: 703 0697 2 ADJ_P2_BUF_DSC: DESCRIPTOR, ! Descriptor for empty P2 buffer.
: 704 0698 2 ADJ_P2_DSC: DESCRIPTOR, ! P2 buffer descriptor
: 705 0699 2 ADJ_P4_BUF: BBLOCK [NML$K_Q10BFLEN],
: 706 0700 2 ADJ_P4_BUF_DSC: DESCRIPTOR, ! P4 buffer descriptor
: 707 0701 2 ADJ_P4_DATA_DSC: DESCRIPTOR, ! P4 buffer data descriptor
: 708 0702 2 ADJ_P4_DATA_PTR, ! P4 buffer data pointer
: 709 0703 2 ADJ_AGENCY_COUNT, ! Number of adjacency entities returned in P4.
: 710 0704 2 MSGSIZE;
: 711 0705 2
: 712 0706 2 ADJ_NFB_DSC [DSC$A_POINTER] = ADJ_NFB_BUF;
: 713 0707 2 ADJ_P2_BUF_DSC [DSC$W_LENGTH] = NML$K_P2BUFLen;
: 714 0708 2 ADJ_P2_BUF_DSC [DSC$A_POINTER] = ADJ_P2_BUF;
: 715 0709 2 ADJ_P4_BUF_DSC [DSC$W_LENGTH] = NML$K_Q10BFLEN;
: 716 0710 2 ADJ_P4_BUF_DSC [DSC$A_POINTER] = ADJ_P4_BUF;
```



```

717 0711 2
718 0712 2 NML$GETINFTABS (.ENTITY,
719 0713 2 .INF,
720 0714 2 ADJ_NFB_DSC,
721 0715 2 ADJ_TABDSC, 1);
722 0716 2
723 0717 2 Build the buffers (NFB, P2, and P4) to get the adjacency information
724 0718 2 for the circuit. If there is a node qualifier, include that as the
725 0719 2 second search key.
726 0720 2
727 0721 2 NML$BLDSHOWBUFS (.ENTITY,
728 0722 2 .ENTITY_LEN,
729 0723 2 .ENTITY_ADDR,
730 0724 2 ADJ_NFB_BUF, ADJ_P2_BUF_DSC, ADJ_P2_DSC,
731 0725 2 .QUAL_PST, .QUAL_LEN, .QUAL_ADR);
732 0726 2
733 0727 2 MSGSIZE = .NICE_MSG_DSC [DSC$W_LENGTH];
734 0728 2 STATUS = 1;
735 0729 2 WHILE .STATUS DO
736 0730 2 BEGIN
737 0731 2
738 0732 2 Get a buffer full of adjacency information for the circuit.
739 0733 2
740 0734 2 STATUS = NML$GETDATA (ADJ_NFB_DSC, ADJ_P2_DSC,
741 0735 2 ADJ_P4_BUF_DSC,
742 0736 2 ADJ_P4_DATA_DSC);
743 0737 2
744 0738 2 IF .STATUS THEN
745 0739 2 BEGIN
746 0740 2 ADJACENCY_COUNT = (.ADJ_P2_DSC [DSC$A_POINTER]);
747 0741 2 ADJ_P4_DATA_PTR = .ADJ_P4_DATA_DSC [DSC$A_POINTER];
748 0742 2
749 0743 2 For each adjacency in the buffer, build a NICE message containing
750 0744 2 the parameters returned in the buffer. Then send the NICE message
751 0745 2 to NCP.
752 0746 2 WHILE (ADJACENCY_COUNT = .ADJACENCY_COUNT - 1) GEQ 0 DO
753 0747 2 BEGIN
754 0748 2
755 0749 2 If this is the first adjacency, include the adjacency info in
756 0750 2 the circuit NICE message already started by the calling routine.
757 0751 2
758 0752 2 IF NOT .NML$B_ADJACENCY_FOUND THEN
759 0753 2 BEGIN
760 0754 2 NML$B_ADJACENCY_FOUND = 1;
761 0755 2 ADJ_P4_DATA_PTR = (.ADJ_P4_DATA_PTR) < 0, 16 > + ! Skip the circuit ID.
762 0756 2 .ADJ_P4_DATA_PTR + 2;
763 0757 2 NML$SHOWPARLIST (NML$GQ_SNDBF_DSC,
764 0758 2 MSGSIZE,
765 0759 2 .ADJ_TABDSC,
766 0760 2 ADJ_P4_DATA_DSC,
767 0761 2 ADJ_P4_DATA_PTR);
768 0762 2 NICE_MSG_DSC [DSC$W_LENGTH] = .MSGSIZE;
769 0763 2 END
770 0764 2 ELSE
771 0765 2
772 0766 2 If the circuit info and the first adjacency info has already been
773 0767 2 returned to NCP, format each of the rest of the adjacencies into a
```



```

: 774      0768 5      ! NICE message of its own without repeating the circuit information
: 775      0769 5      ! except for the circuit ID.
: 776      0770 5      !
: 777      0771 6      BEGIN
: 778      0772 6      NML$PROCESSDATA (.ENTITY,
: 779      0773 6      .ADJ_TABDSC,
: 780      0774 6      ADJ_P4_DATA_DSC,
: 781      0775 6      ADJ_P4_DATA_PTR,
: 782      0776 6      .NICE_MSG_DSC);
: 783      0777 5      END;
: 784      0778 5      NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
: 785      0779 5      .NICE_MSG_DSC [DSC$W_LENGTH]);
: 786      0780 5      END;
: 787      0781 4      END;
: 788      0782 3      END;
: 789      0783 2      END;
: 790      0784 2      !
: 791      0785 2      If the QIO failed for any reason other than end-of-file (no adjacencies were
: 792      0786 2      found), return an error to NCP
: 793      0787 2      IF NOT .STATUS AND
: 794      0788 2      .STATUS NEQ NML$_STS_CMP THEN
: 795      0789 2      BEGIN
: 796      0790 2      NML$BLD REPLY (NML$AB_MSGBLOCK, NICE_MSG_DSC [DSC$W_LENGTH]);
: 797      0791 2      NICE_MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
: 798      0792 2      NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER],
: 799      0793 2      .NICE_MSG_DSC [DSC$W_LENGTH]);
: 800      0794 3      END;
: 801      0795 2      RETURN .STATUS;
: 802      0796 2      ! of NML_SHOW_ADJACENCIES
: 803      0797 1      END;
```

007C 00000 NML_SHOW_ADJACENCIES:

	56	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6	0655	
	55	00000000'	00	9E	00009	MOVAB	NML\$SEND, R6		
	5E	F9B4	CE	9E	00010	MOVAB	NML\$B_ADJACENCY_FOUND, R5		
	FEFC	CD	FF00	CD	9E	00015	MOVAB	-1612(SP), SP	
	FE88	CD	68	8F	9B	0001C	MOVZBW	#104, ADJ_P2_BUF_DSC	0706
	FE8C	CD	FE90	CD	9E	00022	MOVAB	ADJ_P2_BUF, ADJ_P2_BUF_DSC+4	0707
	14	AE	04B0	8F	B0	00029	MOVW	#1200, ADJ_P4_BUF_DSC	0708
	18	AE	1C	AE	9E	0002F	MOVAB	ADJ_P4_BUF, ADJ_P4_BUF_DSC+4	0709
			04	AE	9F	00036	PUSHL	#1	0710
			FEF8	CD	9F	00039	PUSHAB	ADJ_TABDSC	0712
			04	AC	7D	0003D	PUSHAB	ADJ_NFBDSC	
00000000G	7E		00	05	FB	00041	MOVQ	ENTITY, -(SP)	
	7E		18	AC	7D	00048	CALLS	#5, NML\$GETINFTABS	0725
			14	AC	DD	0004C	MOVQ	QUAL_LEN, -(SP)	
			FE80	CD	9F	0004F	PUSHL	QUAL_PST	0721
			FE88	CD	9F	00053	PUSHAB	ADJ_P2_DSC	
			FF00	CD	9F	00057	PUSHAB	ADJ_P2_BUF_DSC	
	7E		0C	AC	7D	0005B	PUSHAB	ADJ_NFB_BUF	0722
			04	AC	DD	0005F	MOVQ	ENTITY_LEN, -(SP)	
						PUSHL	ENTITY	0721	

00000000V	00	09	FB	00062	CALLS	#9, NML\$BLDSHOWBUFS	:	
	52	20	AC	D0	00069	MOVL	NICE_MSG_DSC, R2	0727
04	AE		62	3C	0006D	MOVZWL	(R2), MSGSIZE	
	54		01	D0	00071	MOVL	#1, STATUS	0728
	7B		54	E9	00074	BLBC	STATUS, 5\$	0729
		0C	AE	9F	00077	PUSHAB	ADJ_P4_DATA_DSC	0734
		18	AE	9F	0007A	PUSHAB	ADJ_P4_BUF_DSC	
		FE80	CD	9F	0007D	PUSHAB	ADJ_P2_DSC	
		FEF8	CD	9F	00081	PUSHAB	ADJ_NFBDSC	
00000000V	00	04	FB	00085	CALLS	#4, NML\$GETDATA	:	
	54	50	D0	0008C	MOVL	R0, STATUS	:	
	60	54	E9	0008F	BLBC	STATUS, 5\$:	0737
	53	FE84	DD	D0	00092	MOVL	@ADJ_P2_DSC+4, ADJACENCY_COUNT	0739
08	AE	10	AE	D0	00097	MOVL	ADJ_P4_DATA_DSC+4, ADJ_P4_DATA_PTR	0740
			53	D7	0009C	DECL	ADJACENCY_COUNT	0746
			D4	19	0009E	BLSS	1\$	
	2F		65	E8	000A0	BLBS	NML\$B ADJACENCY FOUND, 3\$	0752
	65		01	90	000A3	MOVB	#1, NML\$B ADJACENCY FOUND	0754
	50	08	BE	3C	000A6	MOVZWL	@ADJ_P4_DATA_PTR, R0	0756
	50	08	AE	C0	000AA	ADDL2	ADJ_P4_DATA_PTR, R0	
08	AE	02	A0	9E	000AE	MOVAB	2(R0), ADJ_P4_DATA_PTR	
		08	AE	9F	000B3	PUSHAB	ADJ_P4_DATA_PTR	0757
		10	AE	9F	000B6	PUSHAB	ADJ_P4_DATA_DSC	
		08	AE	DD	000B9	PUSHL	ADJ_TABDSC	0759
		10	AE	9F	000BC	PUSHAB	MSGSIZE	0757
		00000000G	00	9F	000BF	PUSHAB	NML\$GQ_SNDBFDSC	
00000000G	00	05	FB	000C5	CALLS	#5, NML\$SHOWPARLIST	:	
	62	04	AE	B0	000CC	MOVW	MSGSIZE, (R2)	0762
			15	11	000D0	BRB	4\$	0752
			52	DD	000D2	PUSHL	R2	0776
		0C	AE	9F	000D4	PUSHAB	ADJ_P4_DATA_PTR	0772
		14	AE	9F	000D7	PUSHAB	ADJ_P4_DATA_DSC	
		0C	AE	DD	000DA	PUSHL	ADJ_TABDSC	0773
		04	AC	DD	000DD	PUSHL	ENTITY	0772
00000000V	00	05	FB	000E0	CALLS	#5, NML\$PROCESSDATA	:	
	7E	62	3C	000E7	MOVZWL	(R2), -(SP)	:	0780
		04	A2	DD	000EA	PUSHL	4(R2)	0779
	66	02	FB	000ED	CALLS	#2, NML\$SEND	:	
		AA	11	000F0	BRB	2\$:	0746
FFFFFFFF0	8F	54	D1	000F2	CMPL	STATUS, #-16	:	0789
		20	13	000F9	BEQL	6\$:	
		52	DD	000FB	PUSHL	R2	:	0791
		00	9F	000FD	PUSHAB	NML\$AB_MSGBLOCK	:	
00000000G	00	02	FB	00103	CALLS	#2, NML\$BLD_REPLY	:	
	04	00	9E	0010A	MOVAB	NML\$AB_SNDBUFFER, 4(R2)	:	0792
	7E	62	3C	00112	MOVZWL	(R2), -(SP)	:	0794
		04	A2	DD	00115	PUSHL	4(R2)	0793
	66	02	FB	00118	CALLS	#2, NML\$SEND	:	
	50	54	D0	0011B	MOVL	STATUS, R0	:	0796
		04	0011E	RET			:	0797

; Routine Size: 287 bytes, Routine Base: \$CODE\$ + 03B3

NML\$SHOW
V04-000

NML SHOW parameter module

NML\$SHOW_KNOWN_LOOP Show known loopnode paramet

E 6
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1

Page 28
(8)

```

: 805 0798 1 %SBTTL 'NML$SHOW KNOWN LOOP Show known loopnode parameters'
: 806 0799 1 GLOBAL ROUTINE NML$SHOW_KNOWN_LOOP (ENT, INF, DUM1, DUM2) : NOVALUE =
: 807 0800 1
: 808 0801 1 ++
: 809 0802 1 FUNCTIONAL DESCRIPTION:
: 810 0803 1
: 811 0804 1 This routine reads the volatile data base entries for all
: 812 0805 1 loop nodes.
: 813 0806 1
: 814 0807 1 FORMAL PARAMETERS:
: 815 0808 1
: 816 0809 1 ENT Entity type code.
: 817 0810 1 INF Information type code.
: 818 0811 1 DUM1 Not used.
: 819 0812 1 DUM2 Not used.
: 820 0813 1
: 821 0814 1 --
: 822 0815 1
: 823 0816 2 BEGIN
: 824 0817 2
: 825 0818 2
: 826 0819 2 Counters are not supported for loop nodes.
: 827 0820 2
: 828 0821 2 IF .INF EQLU NML$C_COUNTERS THEN
: 829 0822 2 RETURN;
: 830 0823 2 NML$SHOWMULTIPLE (NML$C_LOOPNODE, .INF, NML$C_ENT_LOO, 0,
: 831 0824 2 0, 0, 0); ! No qualifier
: 832 0825 2
: 833 0826 1 END; ! End of NML$SHOW_KNOWN_LOOP
```

			0000 00000	.ENTRY	NML\$SHOW_KNOWN_LOOP, Save nothing	: 0799
03	08	AC	D1 00002	CMPL	INF, #3	: 0821
		11	13 00006	BEQL	1\$	
		7E	7C 00008	CLRQ	-(SP)	: 0823
		7E	7C 0000A	CLRQ	-(SP)	
7E		03	CE 0000C	MNEGL	#3, -(SP)	
	08	AC	DD 0000F	PUSHL	INF	
		05	DD 00012	PUSHL	#5	
FBC0	CF	07	FB 00014	CALLS	#7, NML\$SHOWMULTIPLE	
		04	00019 1\$:	RET		: 0826

; Routine Size: 26 bytes, Routine Base: \$CODE\$ + 04D2


```
835 0827 1 %SBTTL 'NML$SHOWNODEBYNAME Show volatile node parameters'
836 0828 1 GLOBAL ROUTINE NML$SHOWNODEBYNAME (ENT, INF, LEN, ADR) : NOVALUE =
837 0829 1
838 0830 1 !++
839 0831 1 FUNCTIONAL DESCRIPTION:
840 0832 1
841 0833 1 This routine returns volatile information about the single remote
842 0834 1 node or loop node specified by name.
843 0835 1
844 0836 1 FORMAL PARAMETERS:
845 0837 1
846 0838 1 ENT Entity type code.
847 0839 1 INF Information type code (index).
848 0840 1 LEN Length of entity id string.
849 0841 1 ADR Address of entity id string.
850 0842 1
851 0843 1 !--
852 0844 1
853 0845 2 BEGIN
854 0846 2
855 0847 2 LOCAL
856 0848 2 STATUS,
857 0849 2 P4_DATA_DSC : DESCRIPTOR, ! QIO data descriptor
858 0850 2 P4_DATA_PTR, ! Pointer into P4 buffer
859 0851 2 ENTCODE, ! Internal entity code
860 0852 2 LOOFLAG, ! Loop node flag
861 0853 2 NICE_MSG_DSC : DESCRIPTOR, ! Output message descriptor
862 0854 2 NFBDESC : REF DESCRIPTOR, ! NFB descriptor
863 0855 2 P2DSC : DESCRIPTOR, ! P2 parameter descriptor
864 0856 2 TABDES : REF DESCRIPTOR; ! Information table descriptor
865 0857 2
866 0858 2 NML$GETINFABS (NML$C_NODEBYNAME, .INF, NFBDESC, TABDES, 0);
867 0859 2 NML$BLDP2 (.LEN, .ADR, -1, 0, NML$Q_P2BFDSC, P2DSC);
868 0860 2
869 0861 2 STATUS = NML$GETDATA (.NFBDESC, P2DSC, NML$Q_QIOBFDSC, P4_DATA_DSC);
870 0862 2 IF .STATUS THEN
871 0863 2 BEGIN
872 0864 2 ENTCODE = NML$C_NODEBYNAME;
873 0865 2 P4_DATA_PTR = .P4_DATA_DSC [DSC$A_POINTER];
874 0866 2
875 0867 2 ! If this is a loop node then get different data from NETACP.
876 0868 2 ! The P2 buffer is rebuilt because NETACP returned a collating
877 0869 2 ! value in the P2 buffer from the first QIO - this collating
878 0870 2 ! value will cause NETACP to start looking AFTER the loop node
879 0871 2 ! just found, so it won't find it.
880 0872 2
881 0873 2 LOOFLAG = (.P4_DATA_PTR)<0,32>; ! Get loop node flag
882 0874 2 IF .LOOFLAG NEQ 0 THEN
883 0875 2 BEGIN
884 0876 2 NML$GETINFABS (NML$C_LOOPNODE, .INF, NFBDESC, TABDES, 0);
885 0877 2 NML$BLDP2 (.LEN, .ADR, -1, 0, NML$Q_P2BFDSC, P2DSC);
886 0878 2 STATUS = NML$GETDATA (.NFBDESC, P2DSC, NML$Q_QIOBFDSC, P4_DATA_DSC);
887 0879 2 ENTCODE = NML$C_LOOPNODE; ! Set entity type to loop node
888 0880 2 END
889 0881 2 ELSE
890 0882 2 P4_DATA_PTR = .P4_DATA_PTR + 4; ! Skip over the loop node flag.
891 0883 2 END;
```



```
: 892      0884 2 IF .STATUS THEN
: 893      0885 2     NML$PROCESSDATA (.ENTCODE, .TABDES, P4_DATA_DSC, P4_DATA_PTR, NICE_MSG_DSC)
: 894      0886 2 ELSE
: 895      0887 2     BEGIN
: 896      0888 2     NML$BLD REPLY (NML$AB MSGBLOCK, NICE MSG DSC [DSC$W_LENGTH]);
: 897      0889 2     NICE_MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
: 898      0890 2     END;
: 899      0891 2 NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER], .NICE MSG DSC [DSC$W_LENGTH]);
: 900      0892 1 END;
                        ! End of NML$SHOWNODEBYNAME
```

		01FC 00000	.ENTRY	NML\$SHOWNODEBYNAME, Save R2,R3,R4,R5,R6,R7,-;	
				R8	0828
58	00000000G	00 9E 00002	MOVAB	NML\$GETINFTABS, R8	
57	00000000V	00 9E 00009	MOVAB	NML\$GETDATA, R7	
56	00000000G	00 9E 00010	MOVAB	NML\$GQ QIOBFDSC, R6	
55	00000000G	00 9E 00017	MOVAB	NML\$BLDP2, R5	
54	00000000'	00 9E 0001E	MOVAB	NML\$Q P2BFDSC, R4	
5E		24 C2 00025	SUBL2	#36, SP	
		7E D4 00028	CLRL	-(SP)	0858
	04	AE 9F 0002A	PUSHAB	TABDES	
	0C	AE 9F 0002D	PUSHAB	NFB DSC	
	08	AC DD 00030	PUSHL	INF	
		04 DD 00033	PUSHL	#4	
68		05 FB 00035	CALLS	#5, NML\$GETINFTABS	
	0C	AE 9F 00038	PUSHAB	P2DSC	0859
		54 DD 0003B	PUSHL	R4	
		7E D4 0003D	CLRL	-(SP)	
7E		01 CE 0003F	MNEGL	#1, -(SP)	
7E	0C	AC 7D 00042	MOVQ	LEN, -(SP)	
65		06 FB 00046	CALLS	#6, NML\$BLDP2	
	1C	AE 9F 00049	PUSHAB	P4_DATA_DSC	0861
		56 DD 0004C	PUSHL	R6	
	14	AE 9F 0004E	PUSHAB	P2DSC	
	10	AE DD 00051	PUSHL	NFB DSC	
67		04 FB 00054	CALLS	#4, NML\$GETDATA	
53		50 D0 00057	MOVL	R0, STATUS	
63		53 E9 0005A	BLBC	STATUS, 3\$	0862
52		04 D0 0005D	MOVL	#4, ENT CODE	0864
08	20	AE D0 00060	MOVL	P4_DATA_DSC+4, P4_DATA_PTR	0865
50	08	BE D0 00065	MOVL	@P4_DATA_PTR, LOOFLAG	0873
		37 13 00069	BEQL	1\$	0874
		7E D4 0006B	CLRL	-(SP)	0876
	04	AE 9F 0006D	PUSHAB	TABDES	
	0C	AE 9F 00070	PUSHAB	NFB DSC	
	08	AC DD 00073	PUSHL	INF	
		05 DD 00076	PUSHL	#5	
68		05 FB 00078	CALLS	#5, NML\$GETINFTABS	
	0C	AE 9F 0007B	PUSHAB	P2DSC	0877
		54 DD 0007E	PUSHL	R4	
		7E D4 00080	CLRL	-(SP)	
7E		01 CE 00082	MNEGL	#1, -(SP)	
7E	0C	AC 7D 00085	MOVQ	LEN, -(SP)	
65		06 FB 00089	CALLS	#6, NML\$BLDP2	

NML\$SHOW
V04--000

NML SHOW parameter module
NML\$SHOWNODEBYNAME Show volatile node paramete

H 6
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NMLSHOW.B32;1

Page 31
(9)

		1C	AE	9F	0008C		PUSHAB	P4_DATA_DSC	:	0878
			56	DD	0008F		PUSHL	R6	:	
		14	AE	9F	00091		PUSHAB	P2DSC	:	
		10	AE	DD	00094		PUSHL	NFBDSC	:	
	67		04	FB	00097		CALLS	#4, NML\$GETDATA	:	
	53		50	DD	0009A		MOVL	R0, STATUS	:	
	52		05	DD	0009D		MOVL	#5, ENTCODE	:	0879
			04	11	000A0		BRB	2\$:	0874
08	AE		04	CO	000A2	1\$:	ADDL2	#4, P4_DATA_PTR	:	0882
	17		53	E9	000A6	2\$:	BLBC	STATUS, 3\$:	0884
		14	AE	9F	000A9		PUSHAB	NICE MSG_DSC	:	0885
		0C	AE	9F	000AC		PUSHAB	P4_DATA_PTR	:	
		24	AE	9F	000AF		PUSHAB	P4_DATA_DSC	:	
		0C	AE	DD	000B2		PUSHL	TABDES	:	
			52	DD	000B5		PUSHL	ENTCODE	:	
00000000V	00		05	FB	000B7		CALLS	#5, NML\$PROCESSDATA	:	
			18	11	000BE		BRB	4\$:	
		14	AE	9F	000C0	3\$:	PUSHAB	NICE MSG_DSC	:	0888
		00000000G	00	9F	000C3		PUSHAB	NML\$AB MSGBLOCK	:	
00000000G	00		02	FB	000C9		CALLS	#2, NML\$BLD_REPLY	:	
	18	AE	00	9E	000D0		MOVAB	NML\$AB SNDBUFFER, NICE_MSG_DSC+4	:	0889
	7E		14	AE	3C	4\$:	MOVZWL	NICE_MSG_DSC, -(SP)	:	0891
		1C	AE	DD	000DC		PUSHL	NICE_MSG_DSC+4	:	
00000000G	00		02	FB	000DF		CALLS	#2, NML\$SEND	:	
			04	00	000E6		RET		:	0892

; Routine Size: 231 bytes, Routine Base: \$CODE\$ + 04EC


```
0893 1 %SBTTL 'NML$SHOWEXECUTOR Show volatile executor parameters'
0894 1 GLOBAL ROUTINE NML$SHOWEXECUTOR (ENT, INF, DUM1, DUM2) : NOVALUE =
0895 1
0896 1 ++
0897 1 FUNCTIONAL DESCRIPTION:
0898 1
0899 1 This routine returns volatile information about the executor node.
0900 1
0901 1 FORMAL PARAMETERS:
0902 1
0903 1 ENT Entity type code.
0904 1 INF Information type code (index).
0905 1 DUM1 Not used.
0906 1 DUM2 Not used.
0907 1
0908 1 --
0909 1
0910 2 BEGIN
0911 2
0912 2 LOCAL
0913 2 P4_DATA_DSC : DESCRIPTOR, ! QIO data descriptor
0914 2 P4_DATA_PTR, ! Pointer into P4 buffer
0915 2 DUMDSC : REF DESCRIPTOR, ! Dummy descriptor
0916 2 NICE_MSG_DSC : DESCRIPTOR, ! Output message descriptor
0917 2 NFBDESC : REF DESCRIPTOR, ! NFB descriptor
0918 2 P2DSC : DESCRIPTOR, ! P2 parameter descriptor
0919 2 TABDES : REF DESCRIPTOR; ! Information table descriptor
0920 2
0921 2
0922 2 NML$GETINFRTABS (NML$C_EXECUTOR, .INF, NFBDESC, TABDES, 0);
0923 2
0924 2 NETACP returns all executor node counters from both the executor (LNI)
0925 2 or the remote (NDI) data bases.
0926 2
0927 2 IF .INF NEQ NML$C_COUNTERS THEN
0928 2 BEGIN
0929 2 NML$BLDP2 (-1, 0, -1, 0, NML$Q_P2BFDSC, P2DSC);
0930 2
0931 2 IF NOT NML$GETDATA (.NFBDESC, P2DSC, NML$GQ_EXEBFDSC, NML$GQ_EXEDATDSC)
0932 2 THEN
0933 2 BEGIN
0934 2
0935 2 NML$BLD REPLY (NML$AB MSGBLOCK, NICE_MSG_DSC [DSC$W_LENGTH]);
0936 2 NML$SEND (NML$AB_SNDBUFFER, .NICE_MSG_DSC [DSC$W_LENGTH]);
0937 2 RETURN
0938 2
0939 2 END;
0940 2
0941 2 NML$GL_EXEDATPTR = .NML$GQ_EXEDATDSC [DSC$A_POINTER];
0942 2 NML$GETINFRTABS (NML$C_NODE, .INF, NFBDESC, DUMDSC, 0);
0943 2 END;
0944 2
0945 2 NML$BLDP2 (0, 0, -1, 0, NML$Q_P2BFDSC, P2DSC);
0946 2
0947 2 IF NML$GETDATA (.NFBDESC, P2DSC, NML$GQ_QIOBFDSC, P4_DATA_DSC)
0948 2 THEN
0949 2 BEGIN
```



```
: 959      0950      3
: 960      0951      3      P4_DATA_PTR = .P4_DATA_DSC [DSC$A_POINTER];
: 961      0952      3      NML$PROCESSDATA (NML$C_EXECUTOR, .TABDES, P4_DATA_DSC,
: 962      0953      3      P4_DATA_PTR, NICE_MSG_DSC);
: 963      0954      3
: 964      0955      3      END
: 965      0956      3      ELSE
: 966      0957      3      BEGIN
: 967      0958      3
: 968      0959      3      NML$BLD_REPLY (NML$AB_MSGBLOCK, NICE_MSG_DSC [DSC$W_LENGTH]);
: 969      0960      3      NICE_MSG_DSC [DSC$A_POINTER] = NML$AB_SNDBUFFER;
: 970      0961      3
: 971      0962      3      END;
: 972      0963      3
: 973      0964      3      NML$SEND (.NICE_MSG_DSC [DSC$A_POINTER], .NICE_MSG_DSC [DSC$W_LENGTH]);
: 974      0965      3
: 975      0966      1      END;
:                               ! End of NML$SHOWEXECUTOR
```

		01FC 00000	.ENTRY	NML\$SHOWEXECUTOR, Save R2,R3,R4,R5,R6,R7,R8	: 0894
58	00000000G	00 9E 00002	MOVAB	NML\$GETINF TABS, R8	
57	00000000G	00 9E 00009	MOVAB	NML\$AB_SNDBUFFER, R7	
56	00000000G	00 9E 00010	MOVAB	NML\$BLD_REPLY, R6	
55	00000000G	00 9E 00017	MOVAB	NML\$AB_MSGBLOCK, R5	
54	00000000V	00 9E 0001E	MOVAB	NML\$GETDATA, R4	
53	00000000G	00 9E 00025	MOVAB	NML\$BLDP2, R3	
52	00000000G	00 9E 0002C	MOVAB	NML\$Q_P2BFDSC, R2	
5E		28 C2 00033	SUBL2	#40, SP	
		7E D4 00036	CLRL	-(SP)	: 0922
	04	AE 9F 00038	PUSHAB	TABDES	
	10	AE 9F 0003B	PUSHAB	NFB DSC	
	08	AC DD 0003E	PUSHL	INF	
		07 DD 00041	PUSHL	#7	
68		05 FB 00043	CALLS	#5, NML\$GETINF TABS	
03	08	AC D1 00046	CMPL	INF, #3	: 0927
		55 13 0004A	BEQL	2\$	
	10	AE 9F 0004C	PUSHAB	P2DSC	: 0929
		52 DD 0004F	PUSHL	R2	
		7E D4 00051	CLRL	-(SP)	
7E		01 CE 00053	MNEGL	#1, -(SP)	
		7E D4 00056	CLRL	-(SP)	
7E		01 CE 00058	MNEGL	#1, -(SP)	
63		06 FB 0005B	CALLS	#6, NML\$BLDP2	
	00000000G	00 9F 0005E	PUSHAB	NML\$GQ_EXEDATDSC	: 0931
	00000000G	00 9F 00064	PUSHAB	NML\$GQ_EXEBFDSC	
	18	AE 9F 0006A	PUSHAB	P2DSC	
	14	AE DD 0006D	PUSHL	NFB DSC	
64		04 FB 00070	CALLS	#4, NML\$GETDATA	
10		50 E8 00073	BLBS	R0, 1\$	
	18	AE 9F 00076	PUSHAB	NICE_MSG_DSC	: 0935
		55 DD 00079	PUSHL	R5	
66		02 FB 0007B	CALLS	#2, NML\$BLD_REPLY	
7E	18	AE 3C 0007E	MOVZWL	NICE_MSG_DSC, -(SP)	: 0936
		57 DD 00082	PUSHL	R7	

00000000G	00	00000000G	6E 11 00084	BRB 5\$		
			00 D0 00086 1\$:	MOVL NML\$GQ_EXEDATDSC+4, NML\$GL_EXEDATPTR	:	0941
			7E D4 00091	CLRL -(SP)	:	0942
		08	AE 9F 00093	PUSHAB DUMDSC	:	
		10	AE 9F 00096	PUSHAB NFB DSC	:	
		08	AC DD 00099	PUSHL INF	:	
			03 DD 0009C	PUSHL #3	:	
	68		05 FB 0009E	CALLS #5, NML\$GETINF TABS	:	
		10	AE 9F 000A1 2\$:	PUSHAB P2DSC	:	0945
			52 DD 000A4	PUSHL R2	:	
			7E D4 000A6	CLRL -(SP)	:	
	7E		01 CE 000A8	MNEGL #1, -(SP)	:	
			7E 7C 000AB	CLRL -(SP)	:	
	63		06 FB 000AD	CALLS #6, NML\$BLDP2	:	
		20	AE 9F 000B0	PUSHAB P4_DATA_DSC	:	0947
		00000000G	00 9F 000B3	PUSHAB NML\$GQ_QIOBFDSC	:	
			18 AE 9F 000B9	PUSHAB P2DSC	:	
		14	AE DD 000BC	PUSHL NFB DSC	:	
	64		04 FB 000BF	CALLS #4, NML\$GETDATA	:	
	1C		50 E9 000C2	BLBC R0, 3\$:	
		24	AE D0 000C5	MOVL P4_DATA_DSC+4, P4_DATA_PTR	:	0951
0C	AE		18 AE 9F 000CA	PUSHAB NICE_MSG_DSC	:	0952
		10	AE 9F 000CD	PUSHAB P4_DATA_PTR	:	
		28	AE 9F 000D0	PUSHAB P4_DATA_DSC	:	
		0C	AE DD 000D3	PUSHL TABDES	:	
			07 DD 000D6	PUSHL #7	:	
00000000V	00		05 FB 000D8	CALLS #5, NML\$PROCESSDATA	:	
			0C 11 000DF	BRB 4\$:	0947
		18	AE 9F 000E1 3\$:	PUSHAB NICE_MSG_DSC	:	0959
			55 DD 000E4	PUSHL R5	:	
	66		02 FB 000E6	CALLS #2, NML\$BLD_REPLY	:	
	1C	AE	67 9E 000E9	MOVAB NML\$AB_SNDBUFFER, NICE_MSG_DSC+4	:	0960
		7E	18 AE 3C 000ED 4\$:	MOVZWL NICE_MSG_DSC, -(SP)	:	0964
		20	AE DD 000F1	PUSHL NICE_MSG_DSC+4	:	
00000000G	00		02 FB 000F4 5\$:	CALLS #2, NML\$SEND	:	
			04 000FB	RET	:	0966

; Routine Size: 252 bytes, Routine Base: \$CODE\$ + 05D3


```

: 977 0967 1 %SBTTL 'NML$SHOW MULTIPLE NODES Show multiple node parameters'
: 978 0968 1 GLOBAL ROUTINE NML$SHOW_MULTIPLE_NODES (ENTITY, INF, MULT_TYPE, DUM1,
: 979 0969 1 QUAL_PST, QUAL_LEN, QUAL_ADR) : NOVALUE =
: 980 0970 1
: 981 0971 1 ++
: 982 0972 1 FUNCTIONAL DESCRIPTION:
: 983 0973 1
: 984 0974 1 This routine reads NETACPs volatile data base entries for known
: 985 0975 1 or active nodes.
: 986 0976 1
: 987 0977 1 FORMAL PARAMETERS:
: 988 0978 1 ENTITY Entity ID (Entity Table index)
: 989 0979 1 INF Information type code.
: 990 0980 1 MULT_TYPE NMASC_ENT_KNO => Get KNOWN nodes.
: 991 0981 1 NMASC_ENT_ACT => Get ACTIVE nodes.
: 992 0982 1 DUM1 Dummy parameter. Normally address of entity id string.
: 993 0983 1 QUAL_PST Address of qualifier's entry in the Parameter
: 994 0984 1 Semantic Table (PST).
: 995 0985 1 QUAL_LEN Length of qualifier ID string.
: 996 0986 1 QUAL_ADR Address of qualifier ID string.
: 997 0987 1
: 998 0988 1 SIDE EFFECTS:
: 999 0989 1 Destroys contents of NML$T_LISTBUFFER.
: 1000 0990 1
: 1001 0991 1 --
: 1002 0992 1
: 1003 0993 2 BEGIN
: 1004 0994 2 IF NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] THEN
: 1005 0995 2
: 1006 0996 2 Show the executor node information.
: 1007 0997 2
: 1008 0998 2 NML$SHOWEXECUTOR (NML$C_EXECUTOR, .INF, 0, 0);
: 1009 0999 2
: 1010 1000 2
: 1011 1001 2 Show remote node information.
: 1012 1002 2
: 1013 1003 2 NML$SHOWMULTIPLE (NML$C_NODE, .INF,
: 1014 1004 2 .MULT_TYPE, 0,
: 1015 1005 2 .QUAL_PST, .QUAL_LEN, .QUAL_ADR);
: 1016 1006 2
: 1017 1007 2 IF NOT .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] THEN
: 1018 1008 2
: 1019 1009 2 Show loop node information.
: 1020 1010 2
: 1021 1011 2 NML$SHOW_KNOWN_LOOP (NML$C_LOOPNODE, .INF, 0, 0);
: 1022 1012 2
: 1023 1013 1 END; ! End of NML$SHOW_MULTIPLE_NODES
```

		000C 00000	.ENTRY NML\$SHOW MULTIPLE NODES, Save R2,R3	: 0968
	53 00000000G	00 9E 00002	MOVAB NML\$GL_PRS_FLGS, R3	:
	52 FEF7	CF 9E 00009	MOVAB NML\$SHOWEXECUTOR, R2	:
0A	63	02 E0 0000E	BBS #2, NML\$GL_PRS_FLGS, 1\$: 0994
		7E 7C 00012	CLRQ -(SP)	: 0998

NML\$SHOW
V04-000

NML SHOW parameter module
NML\$SHOW_MULTIPLE_NODES

Show multiple node par

M 6
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1

Page 36
(11)

		08	AC	DD	00014	PUSHL	INF	:	
			07	DD	00017	PUSHL	#7	:	
	62		04	FB	00019	CALLS	#4, NML\$SHOWEXECUTOR	:	
	7E	18	AC	7D	0001C	MOVQ	QUAL_LEN, -(SP)	:	1005
		14	AC	DD	00020	PUSHL	QUAL_PST	:	
			7E	D4	00023	CLRL	-(SP)	:	1003
	7E	08	AC	7D	00025	MOVQ	INF, -(SP)	:	
			03	DD	00029	PUSHL	#3	:	
	FAD8	C2	07	FB	0002B	CALLS	#7, NML\$SHOWMULTIPLE	:	
0C		63	02	E0	00030	BBS	#2, NML\$GL_PRS_FLGS, 2\$:	1007
			7E	7C	00034	CLRQ	-(SP)	:	1011
		08	AC	DD	00036	PUSHL	INF	:	
			05	DD	00039	PUSHL	#5	:	
	FEFF	C2	04	FB	0003B	CALLS	#4, NML\$SHOW_KNOWN_LOOP	:	
			04	00040	2\$:	RET		:	1013

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 06CF


```
1025 1014 1 %SBTTL 'NML$GET_ENTITY_IDS Get multiple entities'
1026 1015 1 GLOBAL ROUTINE NML$GET_ENTITY_IDS (ENTITY, ENTITY_LEN, ENTITY_ADR,
1027 1016 1 SHOW_STARTED, LISDSC) =
1028 1017 1
1029 1018 1 ++
1030 1019 1 FUNCTIONAL DESCRIPTION:
1031 1020 1
1032 1021 1 This routine is called for doing SET commands to get the Entity
1033 1022 1 IDs to return in the NICE response messages for each entity updated.
1034 1023 1 On the first call (when SHOW_STARTED is false), this routine
1035 1024 1 sets up the QIO buffers to get the IDs of the entities
1036 1025 1 in the specified ACP database. On all calls, this routine
1037 1026 1 issues the SHOW QIO to get a buffer of entity IDs.
1038 1027 1
1039 1028 1 FORMAL PARAMETERS:
1040 1029 1
1041 1030 1 ENTITY Internal entity type code.
1042 1031 1 ENTITY_LEN NMASC_ENT_KNO => Get KNOWN entries of entity.
1043 1032 1 >0 Get all entries of specified entity (which
1044 1033 1 is qualified and therefore has multiple entries).
1045 1034 1 ENTITY_ADR Address of entity ID string.
1046 1035 1 SHOW_STARTED FALSE=>start at beginning of ACPs database.
1047 1036 1 LISDSC Address of longword to get list descriptor
1048 1037 1 address.
1049 1038 1
1050 1039 1 ROUTINE VALUE:
1051 1040 1 COMPLETION CODES:
1052 1041 1
1053 1042 1 If the descriptor is found for the specified entity then success
1054 1043 1 (NML$STS_SUC) is returned. If the end of the data base has been
1055 1044 1 reached then an error is returned (NML$STS_CMP). If any other
1056 1045 1 error is encountered then a message is signalled.
1057 1046 1
1058 1047 1 SIDE EFFECTS:
1059 1048 1
1060 1049 1 NONE
1061 1050 1
1062 1051 1 --
1063 1052 1
1064 1053 2 BEGIN
1065 1054 2
1066 1055 2 Canned NFBs to get KNOWN entities.
1067 1056 2
1068 1057 2
1069 P 1058 2 $NFBDS (KNO_CIR_NFBDS, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
1070 P 1059 2 CIR, NMESC_CIRCUITS
1071 P 1060 2 NFB$C_WILDCARD,, Search key 1 = wildcard, oper1 = eql
1072 P 1061 2 NFB$C_WILDCARD,, Search key 2 = wildcard, oper2 = eql
1073 1062 2 NAM);
1074 P 1063 2 $NFBDS (KNO_LIN_NFBDS, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
1075 P 1064 2 PLI, NMESC_LINE
1076 P 1065 2 NFB$C_WILDCARD,, Search key 1 = wildcard, oper1 = eql
1077 P 1066 2 NFB$C_WILDCARD,, Search key 2 = wildcard, oper2 = eql
1078 1067 2 NAM);
1079 P 1068 2 $NFBDS (KNO_SNK_NFBDS, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
1080 P 1069 2 ESI, NMESC_SINK
1081 P 1070 2 NFB$C_WILDCARD,, Search key 1 = wildcard, oper1 = eql
```



```
: 1082 P 1071 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1083 P 1072 2 SNFB$C (KNO_LOG_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1084 P 1073 2 EFI, ! NML$C_LOGGING
: 1085 P 1074 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1086 P 1075 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1087 P 1076 2 SIN);
: 1088 P 1077 2 SNFB$C (KNO_OBJ_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1089 P 1078 2 OBI, ! NML$C_OBJECT
: 1090 P 1079 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1091 P 1080 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1092 P 1081 2 NAM);
: 1093 P 1082 2 SNFB$C (KNO_LOO_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1094 P 1083 2 NDI, ! NML$C_LOOPNODE
: 1095 P 1084 2 LOO, ! Search key 1 = loopnode, oper1 = eql
: 1096 P 1085 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1097 P 1086 2 NNA);
: 1098 P 1087 2 SNFB$C (KNO_NOD_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1099 P 1088 2 NDI, ! NML$C_NODE
: 1100 P 1089 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1101 P 1090 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1102 P 1091 2 LOO,ADD,NNA);
: 1103 P 1092 2 SNFB$C (KNO_ACC_NET_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1104 P 1093 2 XAI, ! NML$C_PROT_DTE
: 1105 P 1094 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1106 P 1095 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1107 P 1096 2 NET);
: 1108 P 1097 2 SNFB$C (KNO_DTE_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1109 P 1098 2 XDI, ! NML$C_PROT_DTE
: 1110 P 1099 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1111 P 1100 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1112 P 1101 2 DTE);
: 1113 P 1102 2 SNFB$C (KNO_GRP_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1114 P 1103 2 XGI, ! NML$C_PROT_GRP
: 1115 P 1104 2 GRP, ! Search key 1 = group name, oper1 = eql
: 1116 P 1105 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1117 P 1106 2 GRP);
: 1118 P 1107 2 SNFB$C (KNO_X25_DST_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1119 P 1108 2 XD5, ! NML$C_X25_SERV_DEST
: 1120 P 1109 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1121 P 1110 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1122 P 1111 2 DST);
: 1123 P 1112 2 SNFB$C (KNO_X25_TRPNT_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1124 P 1113 2 XTT, ! NML$C_TRACEPNT
: 1125 P 1114 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1126 P 1115 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1127 P 1116 2 TPT);
: 1128 P 1117 2 SNFB$C (KNO_X29_DST_NFB$C, SHOW, NFB$M_MULT OR NFB$M_ERRUPD,
: 1129 P 1118 2 XD9, ! NML$C_X29_SERV_DEST
: 1130 P 1119 2 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql
: 1131 P 1120 2 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql
: 1132 P 1121 2 DST);
: 1133 P 1122 2
: 1134 P 1123 2
: 1135 P 1124 2
: 1136 P 1125 2
: 1137 P 1126 2 ! NFBs to get ACTIVE entries (used only for logging database. Other
: 1138 P 1127 2 ! entities use NML$SHOWMULTIPLE.
```



```
: 1139
: 1140
: 1141
: 1142
: 1143
: 1144
: 1145
: 1146
: 1147
: 1148
: 1149
: 1150
: 1151
: 1152
: 1153
: 1154
: 1155
: 1156
: 1157
: 1158
: 1159
: 1160
: 1161
: 1162
: 1163
: 1164
: 1165
: 1166
: 1167
: 1168
: 1169
: 1170
: 1171
: 1172
: 1173
: 1174
: 1175
: 1176
: 1177
: 1178
: 1179
: 1180
: 1181
: 1182
: 1183
: 1184
: 1185
: 1186
: 1187
: 1188
: 1189
: 1190
: 1191
: 1192
: 1193
: 1194
: 1195

P 1128 2 !
P 1129 $NFBDS (ACT SNK NFBDS, SHOW, NFB$M_MULT OR NFB$M_ERRUPD, ESI,
P 1130 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql.
P 1131 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql.
1132 SNK, STA);
1133
P 1134 $NFBDS (ACT LOG NFBDS, SHOW, NFB$M_MULT OR NFB$M_ERRUPD, EFI,
P 1135 NFB$C_WILDCARD,, ! Search key 1 = wildcard, oper1 = eql.
P 1136 NFB$C_WILDCARD,, ! Search key 2 = wildcard, oper2 = eql.
1137 SIN);
1138
1139
1140 OWN
1141 NFBDS : REF DESCRIPTOR,
1142 P2_BUF : VECTOR [NML$K_P2BUFLN],
1143 P2DSC : DESCRIPTOR;
1144
1145 BIND
1146 P2_BUF_DSC = UPLIT (NML$K_P2BUFLN, P2_BUF) : DESCRIPTOR;
1147
1148 LOCAL
1149 MSGSIZE,
1150 RESLEN : WORD,
1151 STATUS,
1152 SRCHLEN1,
1153 SRCHADR1,
1154 SRCHLEN2,
1155 SRCHADR2,
1156 NFB: REF BBLOCK;
1157
1158
1159 To do the QIO, three buffers are needed:
1160 The NFB which tells NETACP which database to access and what
1161 parameters to return.
1162 The P2 buffer which tells NETACP which entity to return the
1163 data for.
1164 The P4 buffer in which NETACP returns the requested data.
1165 If this is the first call on NML$GET_ENTITY_IDS for the operation,
1166 set up the start key, if there is one, and build the P2 buffer for the SHOW
1167 QIO. The ACP writes a value into the P2 buffer so that, when the next SHOW
1168 QIO is issued, it knows how far in its database it got on the last call.
1169 This way a buffer full of entity IDs is returned on each call, and subsequent
1170 calls return the next batch of entity IDs. Thus, the P2 buffer only needs
1171 to be built once for each operation, and is used for multiple
1172 calls until all entities in the database have been returned.
1173
1174 IF NOT .SHOW_STARTED THEN
1175 BEGIN
1176 SRCHLEN1 = -1;
1177 SRCHADR1 = 0;
1178 SRCHLEN2 = -1;
1179 SRCHADR2 = 0;
1180
1181 IF .ENTITY_LEN EQL NML$C_ENT_ACT THEN
1182
1183 Set up to get ACTIVE entity entries.
1184
```



```
: 1196      1185  4      BEGIN
: 1197      1186  4      SELECTONEU .ENTITY OF
: 1198      1187  4      SET
: 1199      1188  4      [NML$C_SINK]: NFBDSK = ACT_SNK_NFBDSK;
: 1200      1189  4      [NML$C_LOGGING]: NFBDSK = ACT_LOG_NFBDSK;
: 1201      1190  4      TES
: 1202      1191  4      END
: 1203      1192  3      ELSE
: 1204      1193  4      BEGIN
: 1205      1194  4      : Use canned NFBs (above) and build a P2 buffer to get KNOWN entity entries.
: 1206      1195  4      :
: 1207      1196  4      :
: 1208      1197  4      SELECTONEU .ENTITY OF
: 1209      1198  4      SET
: 1210      1199  4      [NML$C_CIRCUIT]: NFBDSK = KNO_CIR_NFBDSK;      ! Circuits
: 1211      1200  4      [NML$C_LINE]: NFBDSK = KNO_LIN_NFBDSK;      ! Lines
: 1212      1201  4      [NML$C_SINK]: NFBDSK = KNO_SNK_NFBDSK;      ! Logging (sinks)
: 1213      1202  4      [NML$C_LOGGING]: NFBDSK = KNO_LOG_NFBDSK;      ! Logging (filters)
: 1214      1203  4      [NML$C_LOOPNODE]: NFBDSK = KNO_LOOP_NFBDSK;      ! Loop nodes
: 1215      1204  5      BEGIN
: 1216      1205  5      NFBDSK = KNO_LOO_NFBDSK;
: 1217      1206  5      SRCHLEN1 = 0;
: 1218      1207  5      SRCHADR1 = 1;      ! Match loop nodes
: 1219      1208  4      END;
: 1220      1209  4      [NML$C_OBJECT]: NFBDSK = KNO_OBJ_NFBDSK;      ! Objects
: 1221      1210  4      [NML$C_NODE]: NFBDSK = KNO_NOD_NFBDSK;      ! Remote nodes
: 1222      1211  4      [NML$C_X25_ACCESS]: NFBDSK = KNO_ACC_NET_NFBDSK;      ! X-25 Access Network
: 1223      1212  4      [NML$C_PROT_DTE]: NFBDSK = KNO_DTE_NFBDSK;      ! X-25 Protocol DTE
: 1224      1213  4      [NML$C_PROT_GRP]:
: 1225      1214  4      :
: 1226      1215  4      : GROUPS have one database entry for each DTE in the group.
: 1227      1216  4      : If working with a specific group, get all the entries for
: 1228      1217  4      : the specified group. Otherwise, get all entries for all
: 1229      1218  4      : groups.
: 1230      1219  4      :
: 1231      1220  4      BEGIN
: 1232      1221  5      NFBDSK = KNO_GRP_NFBDSK;
: 1233      1222  5      NFB = NFBDSK [DSC$A_POINTER];
: 1234      1223  5      IF .ENTITY_LEN GTR 0 THEN
: 1235      1224  5      BEGIN
: 1236      1225  6      NFB [NFB$S_SRCH_KEY] = NFB$C_XGI_GRP;
: 1237      1226  6      SRCHLEN1 = .ENTITY_LEN;
: 1238      1227  6      SRCHADR1 = .ENTITY_ADR;
: 1239      1228  6      END
: 1240      1229  6      ELSE
: 1241      1230  5      NFB [NFB$S_SRCH_KEY] = NFB$C_WILDCARD;
: 1242      1231  5      END;
: 1243      1232  4      [NML$C_X25_SERV_DEST]:
: 1244      1233  4      NFBDSK = KNO_X25_DST_NFBDSK;      ! X-25 Server Destination
: 1245      1234  4      [NML$C_TRACEPNT]:
: 1246      1235  4      NFBDSK = KNO_X25_TRPNT_NFBDSK;      ! X-25 Tracepoint
: 1247      1236  4      [NML$C_X29_SERV_DEST]:
: 1248      1237  4      NFBDSK = KNO_X29_DST_NFBDSK;      ! X-29 Server Destination
: 1249      1238  4      [NML$C_LINKS]: ;      ! Logical links don't use this.
: 1250      1239  4      [OTHERWISE]:
: 1251      1240  4      RETURN NML$STS_MPR;
: 1252      1241  4
```



```
1253 1242 4      TES;
1254 1243      END;
1255 1244
1256 1245      | Build the P2 QIO buffer.
1257 1246      |
1258 1247      | NML$BLDP2 ( .SRCHLEN1, .SRCHADR1,
1259 1248      |      .SRCHLEN2, .SRCHLEN2,
1260 1249      |      P2_BUF_DSC, P2DSC);
1261 1250
1262 1251      END;
1263 1252
1264 1253      | Get a bufferfull of entities. Calling routine must reenter this routine
1265 1254      | to get subsequent bufferfulls.
1266 1255
1267 1256      STATUS = NML$GETDATA (.NFB DSC, P2DSC, NML$Q_LISTBFDSC, .LISDSC);
1268 1257
1269 1258      | If the error returned is NML$STS_CMP then the end of the data base
1270 1259      | has been reached. If any other error is returned then build the
1271 1260      | appropriate message and signal it.
1272 1261
1273 1262      IF NOT .STATUS AND (.STATUS NEQ NML$STS_CMP)
1274 1263      THEN
1275 1264          BEGIN
1276 1265              NML$BLD_REPLY (NML$AB MSGBLOCK, MSGSIZE);
1277 1266              $SIGNAL_MSG (NML$AB_SNDBUFFER, .MSGSIZE);
1278 1267          END;
1279 1268
1280 1269      RETURN .STATUS
1281 1270
1282 1271      ! End of NML$GET_ENTITY_IDS
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

```
0000001C 00020 P.AAE: .LONG 28
00000000 00024 .ADDRESS U.3
0000001C 00028 P.AAF: .LONG 28
00000000 0002C .ADDRESS U.5
0000001C 00030 P.AAG: .LONG 28
00000000 00034 .ADDRESS U.7
0000001C 00038 P.AAH: .LONG 28
00000000 0003C .ADDRESS U.9
0000001C 00040 P.AAI: .LONG 28
00000000 00044 .ADDRESS U.11
0000001C 00048 P.AAJ: .LONG 28
00000000 0004C .ADDRESS U.13
00000024 00050 P.AAK: .LONG 36
00000000 00054 .ADDRESS U.15
0000001C 00058 P.AAL: .LONG 28
00000000 0005C .ADDRESS U.17
0000001C 00060 P.AAM: .LONG 28
00000000 00064 .ADDRESS U.19
0000001C 00068 P.AAN: .LONG 28
00000000 0006C .ADDRESS U.21
0000001C 00070 P.AAO: .LONG 28
00000000 00074 .ADDRESS U.23
```



```
0000001C 00078 P.AAP: .LONG 28
00000000 0007C .ADDRESS U.25
0000001C 00080 P.AAQ: .LONG 28
00000000 00084 .ADDRESS U.27
00000020 00088 P.AAR: .LONG 32
00000000 0008C .ADDRESS U.29
0000001C 00090 P.AAS: .LONG 28
00000000 00094 .ADDRESS U.31
00000068 00098 P.AAT: .LONG 104
00000000 0009C .ADDRESS P2_BUF

.PSECT $OWNS$,NOEXE,2

22 006E8 : NFB
      U.3: .BYTE 34
03 006E9 .BYTE 3
04 006EA .BYTE 4
00 006EB .BYTE 0
00000001 006EC .LONG 1
00000001 006F0 .LONG 1
00 006F4 .BYTE 0
00 006F5 .BYTE 0
0000 006F6 .WORD 0
04020041 006F8 .LONG 67240001
00000000 006FC .LONG 0
      00700 .BLKB 4
22 00704 : NFB
      U.5: .BYTE 34
03 00705 .BYTE 3
05 00706 .BYTE 5
00 00707 .BYTE 0
00000001 00708 .LONG 1
00000001 0070C .LONG 1
00 00710 .BYTE 0
00 00711 .BYTE 0
0000 00712 .WORD 0
05020041 00714 .LONG 84017217
00000000 00718 .LONG 0
      0071C .BLKB 4
22 00720 : NFB
      U.7: .BYTE 34
03 00721 .BYTE 3
07 00722 .BYTE 7
00 00723 .BYTE 0
00000001 00724 .LONG 1
00000001 00728 .LONG 1
00 0072C .BYTE 0
00 0072D .BYTE 0
0000 0072E .WORD 0
07010010 00730 .LONG 117506064
00000000 00734 .LONG 0
      00738 .BLKB 4
22 0073C : NFB
      U.9: .BYTE 34
03 0073D .BYTE 3
06 0073E .BYTE 6
00 0073F .BYTE 0
```


00000001	00740	.LONG	1	
00000001	00744	.LONG	1	
00	00748	.BYTE	0	
00	00749	.BYTE	0	
0000	0074A	.WORD	0	
06010010	0074C	.LONG	100728848	
00000000	00750	.LONG	0	
	00754	.BLKB	4	
22	00758	: NFB U-11:		
03	00759	.BYTE	34	
03	0075A	.BYTE	3	
00	0075B	.BYTE	0	
00000001	0075C	.LONG	1	
00000001	00760	.LONG	1	
00	00764	.BYTE	0	
00	00765	.BYTE	0	
0000	00766	.WORD	0	
03020044	00768	.LONG	50462788	
00000000	0076C	.LONG	0	
	00770	.BLKB	4	
22	00774	: NFB U-13:		
03	00775	.BYTE	34	
02	00776	.BYTE	3	
00	00777	.BYTE	2	
02000002	00778	.LONG	33554434	
00000001	0077C	.LONG	1	
00	00780	.BYTE	0	
00	00781	.BYTE	0	
0000	00782	.WORD	0	
02020043	00784	.LONG	33685571	
00000000	00788	.LONG	0	
	0078C	.BLKB	4	
22	00790	: NFB U-15:		
03	00791	.BYTE	34	
02	00792	.BYTE	3	
00	00793	.BYTE	2	
00000001	00794	.BYTE	0	
00000001	00798	.LONG	1	
00	0079C	.LONG	1	
00	0079D	.BYTE	0	
0000	0079E	.BYTE	0	
02000002	007A0	.WORD	0	
02010012	007A4	.LONG	33554434	
02020043	007A8	.LONG	33619986	
00000000	007AC	.LONG	33685571	
	007B0	.LONG	0	
22	007B4	.BLKB	4	
		: NFB U-17:		
03	007B5	.BYTE	34	
1B	007B6	.BYTE	3	
00	007B7	.BYTE	27	
00000001	007B8	.BYTE	0	
00000001	007B8	.LONG	1	
00	007BC	.LONG	1	
	007C0	.BYTE	0	

00	007C1	.BYTE	0	:
0000	007C2	.WORD	0	:
1B020041	007C4	.LONG	453115969	:
00000000	007C8	.LONG	0	:
	007CC	.BLKB	4	:
22	007D0	: NFB		:
	U:19:			:
03	007D1	.BYTE	34	:
0B	007D2	.BYTE	3	:
00	007D3	.BYTE	11	:
00000001	007D4	.BYTE	0	:
00000001	007D8	.LONG	1	:
00	007DC	.LONG	1	:
00	007DD	.BYTE	0	:
0000	007DE	.BYTE	0	:
0B020041	007E0	.WORD	0	:
00000000	007E4	.LONG	184680513	:
	007E8	.LONG	0	:
22	007EC	.BLKB	4	:
	U:21:			:
03	007ED	.BYTE	34	:
0A	007EE	.BYTE	3	:
00	007EF	.BYTE	10	:
0A020041	007F0	.BYTE	0	:
00000001	007F4	.LONG	167903297	:
00	007F8	.LONG	1	:
00	007F9	.BYTE	0	:
0000	007FA	.BYTE	0	:
0A020041	007FC	.WORD	0	:
00000000	00800	.LONG	167903297	:
	00804	.LONG	0	:
22	00808	.BLKB	4	:
	U:23:			:
03	00809	.BYTE	34	:
0D	0080A	.BYTE	3	:
00	0080B	.BYTE	13	:
00000C01	0080C	.BYTE	0	:
00000001	00810	.LONG	1	:
00	00814	.LONG	1	:
00	00815	.BYTE	0	:
0000	00816	.BYTE	0	:
0D020041	00818	.WORD	0	:
00000000	0081C	.LONG	218234945	:
	00820	.LONG	0	:
22	00824	.BLKB	4	:
	U:25:			:
03	00825	.BYTE	34	:
11	00826	.BYTE	3	:
00	00827	.BYTE	17	:
00000001	00828	.BYTE	0	:
00000001	0082C	.LONG	1	:
00	00830	.LONG	1	:
00	00831	.BYTE	0	:
0000	00832	.BYTE	0	:
11020041	00834	.WORD	0	:
00000000	00838	.LONG	285343809	:
	0083C	.LONG	0	:
		.BLKB	4	:

22	00840	: NFB	
	U.27:		
03	00841	.BYTE	34
0F	00842	.BYTE	3
00	00843	.BYTE	15
00000001	00844	.BYTE	0
00000001	00848	.LONG	1
00	0084C	.LONG	1
00	0084D	.BYTE	0
0000	0084E	.BYTE	0
0F020041	00850	.WORD	0
00000000	00854	.LONG	251789377
	00858	.LONG	0
22	0085C	.BLKB	4
	: NFB		
	U.29:		
03	0085D	.BYTE	34
07	0085E	.BYTE	3
00	0085F	.BYTE	7
00000001	00860	.BYTE	0
00000001	00864	.LONG	1
00	00868	.LONG	1
00	00869	.BYTE	0
0000	0086A	.BYTE	0
07010010	0086C	.WORD	0
07010011	00870	.LONG	117506064
00000000	00874	.LONG	117506065
	00878	.LONG	0
22	0087C	.BLKB	4
	: NFB		
	U.31:		
03	0087D	.BYTE	34
06	0087E	.BYTE	3
00	0087F	.BYTE	6
00000001	00880	.BYTE	0
00000001	00884	.LONG	1
00	00888	.LONG	1
00	00889	.BYTE	0
0000	0088A	.BYTE	0
06010010	0088C	.WORD	0
00000000	00890	.LONG	100728848
	00894	.LONG	0
	00898	.BLKB	4
	NFBDSC:	.BLKB	4
	P2 BUF:	.BLKB	416
00A3C	P2DSC:	.BLKB	8

U.4=	P.AAE
U.6=	P.AAF
U.8=	P.AAG
U.10=	P.AAH
U.12=	P.AAI
U.14=	P.AAJ
U.16=	P.AAK
U.18=	P.AAL
U.20=	P.AAM
U.22=	P.AAN
U.24=	P.AAO
U.26=	P.AAP
U.28=	P.AAQ

U.30=
U.32=
P2_BUF_DSC=P.AAR
P.AAS
P.AAT

					.PSECT	\$CODE\$,NOWRT,2	
			003C	00000	.ENTRY	NML\$GET_ENTITY_IDS, Save R2,R3,R4,R5	1015
55	00000000'	00	9E	00002	MOVAB	U.30, R5	
54	00000000'	00	9E	00009	MOVAB	NFBDS, R4	
5E		04	C2	00010	SUBL2	#4, SP	
03	10	AC	E9	00013	BLBC	SHOW_STARTED, 1\$	1174
		00FB	31	00017	BRW	25\$	
52		01	CE	0001A	MNEGL	#1, SRCHLEN1	1176
53		01	CE	0001D	MNEGL	#1, SRCHLEN2	1178
		50	7C	00020	CLRQ	SRCHADR2	1179
50	04	AC	D0	00022	MOVL	ENTITY, R0	1186
8F	08	AC	D1	00026	CMPL	ENTITY_LEN, #-2	1181
		15	12	0002E	BNEQ	3\$	
02		50	D1	00030	CMPL	R0, #2	1188
		05	12	00033	BNEQ	2\$	
64		65	9E	00035	MOVAB	ACT_SNK_NFBDS, NFBDS	
		6E	11	00038	BRB	12\$	
01		50	D1	0003A	CMPL	R0, #1	1189
		69	12	0003D	BNEQ	12\$	
64	08	A5	9E	0003F	MOVAB	ACT_LOG_NFBDS, NFBDS	
		63	11	00043	BRB	12\$	1185
09		50	D1	00045	CMPL	R0, #9	1199
		06	12	00048	BNEQ	4\$	
64	98	A5	9E	0004A	MOVAB	KNO_CIR_NFBDS, NFBDS	
		7F	11	0004E	BRB	14\$	
		50	D5	00050	TSTL	R0	1200
		06	12	00052	BNEQ	5\$	
64	A0	A5	9E	00054	MOVAB	KNO_LIN_NFBDS, NFBDS	
		7B	11	00058	BRB	16\$	
02		50	D1	0005A	CMPL	R0, #2	1201
		06	12	0005D	BNEQ	6\$	
64	A8	A5	9E	0005F	MOVAB	KNO_SNK_NFBDS, NFBDS	
		7B	11	00063	BRB	18\$	
01		50	D1	00065	CMPL	R0, #1	1202
		06	12	00068	BNEQ	7\$	
64	B0	A5	9E	0006A	MOVAB	KNO_LOG_NFBDS, NFBDS	
		7B	11	0006E	BRB	20\$	
05		50	D1	00070	CMPL	R0, #5	1203
		09	12	00073	BNEQ	8\$	
64	C0	A5	9E	00075	MOVAB	KNO_LOO_NFBDS, NFBDS	1205
51		01	7D	00079	MOVQ	#1, SRCHADR1	1207
		78	11	0007C	BRB	22\$	1197
08		50	D1	0007E	CMPL	R0, #8	1209
		06	12	00081	BNEQ	9\$	
64	B8	A5	9E	00083	MOVAB	KNO_OBJ_NFBDS, NFBDS	
		78	11	00087	BRB	24\$	
03		50	D1	00089	CMPL	R0, #3	1210
		06	12	0008C	BNEQ	10\$	
64	C8	A5	9E	0008E	MOVAB	KNO_NOD_NFBDS, NFBDS	
		6D	11	00092	BRB	24\$	
0D		50	D1	00094	CMPL	R0, #13	1211

		06	12	00097	BNEQ	11\$:	
	64	D0	A5	9E 00099	MOVAB	KNO_ACC_NET_NFBDSC, NFBDSC	:	1212
			62	11 0009D	BRB	24\$:	
	0F		50	D1 0009F 11\$:	CMPL	R0, #15	:	1213
			06	12 000A2	BNEQ	13\$:	
	64	D8	A5	9E 000A4	MOVAB	KNO_DTE_NFBDSC, NFBDSC	:	
			57	11 000A8 12\$:	BRB	24\$:	
	10		50	D1 000AA 13\$:	CMPL	R0, #16	:	1214
			28	12 000AD	BNEQ	17\$:	
	64	E0	A5	9E 000AF	MOVAB	KNO_GRP_NFBDSC, NFBDSC	:	1222
	50		64	D0 000B3	MOVL	NFBDSC, R0	:	1223
	50	04	A0	D0 000B6	MOVL	4(R0), NFB	:	
		08	AC	D5 000BA	TSTL	ENTITY_LEN	:	1224
			12	15 000BD	BLEQ	15\$:	
04	A0	0A020041	8F	D0 000BF	MOVL	#167903297, 4(NFB)	:	1226
	52	08	AC	D0 000C7	MOVL	ENTITY_LEN, SRCHLEN1	:	1227
	51	0C	AC	D0 000CB	MOVL	ENTITY_ADR, SRCHADR1	:	1228
			30	11 000CF 14\$:	BRB	24\$:	1224
04	A0		01	D0 000D1 15\$:	MOVL	#1, 4(NFB)	:	1231
			2A	11 000D5 16\$:	BRB	24\$:	1197
	12		50	D1 000D7 17\$:	CMPL	R0, #18	:	1233
			06	12 000DA	BNEQ	19\$:	
	64	E8	A5	9E 000DC	MOVAB	KNO_X25_DST_NFBDSC, NFBDSC	:	1234
			1F	11 000E0 18\$:	BRB	24\$:	
	14		50	D1 000E2 19\$:	CMPL	R0, #20	:	1235
			06	12 000E5	BNEQ	21\$:	
	64	F0	A5	9E 000E7	MOVAB	KNO_X25_TRPNT_NFBDSC, NFBDSC	:	1236
			14	11 000EB 20\$:	BRB	24\$:	
	16		50	D1 000ED 21\$:	CMPL	R0, #22	:	1237
			06	12 000F0	BNEQ	23\$:	
	64	F8	A5	9E 000F2	MOVAB	KNO_X29_DST_NFBDSC, NFBDSC	:	1238
			09	11 000F6 22\$:	BRB	24\$:	
	18		50	D1 000F8 23\$:	CMPL	R0, #24	:	1239
			04	13 000FB	BEQL	24\$:	
	50		0A	CE 000FD	MNEGL	#10, R0	:	1241
			04	00100	RET		:	
		01A4	C4	9F 00101 24\$:	PUSHAB	P2DSC	:	1248
	10		A5	9F 00105	PUSHAB	P2_BUF_DSC	:	
			53	DD 00108	PUSHL	SRCHLEN2	:	1249
			0A	BB 0010A	PUSHR	#^M<R1,R3>	:	1248
			52	DD 0010C	PUSHL	SRCHLEN1	:	
00000000G	00		06	FB 0010E	CALLS	#6, NML\$BLDP2	:	
		14	AC	DD 00115 25\$:	PUSHL	LISDSC	:	1256
		FF78	C5	9F 00118	PUSHAB	NML\$Q_LISTBFDSC	:	
		01A4	C4	9F 0011C	PUSHAB	P2DSC	:	
			64	DD 00120	PUSHL	NFBDSC	:	
00000000V	00		04	FB 00122	CALLS	#4, NML\$GETDATA	:	
	52		50	D0 00129	MOVL	R0, STATUS	:	
	2D		52	E8 0012C	BLBS	STATUS, 26\$:	1262
FFFFFFFF0	8F		52	D1 0012F	CMPL	STATUS, #-16	:	
			24	13 00136	BEQL	26\$:	
			5E	DD 00138	PUSHL	SP	:	1265
		00000000G	00	9F 0013A	PUSHAB	NML\$AB_MSGBLOCK	:	
00000000G	00		02	FB 00140	CALLS	#2, NML\$BLD_REPLY	:	
			6E	DD 00147	PUSHL	MSGSIZE	:	1266
		00000000G	00	9F 00149	PUSHAB	NML\$AB_SNDBUFFER	:	
		01F90000	8F	DD 0014F	PUSHL	#33095680	:	

NML\$SHOW
VO4-000

NML SHOW parameter module
NML\$GET_ENTITY_IDS Get multiple entities

L 7
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VM\$MASTER:[NML.SRC]NML\$SHOW.B32;1 Page 48
(12)

00000000G 00
50

03 FB 00155
52 D0 0015C 26\$:
04 0015F

CALLS #3, LIB\$SIGNAL
MOVL STATUS, R0
RET

:
: 1269
: 1271

; Routine Size: 352 bytes, Routine Base: \$CODE\$ + 0710


```
: 1284      1272 1 %SBTTL 'NML$BLDSHOWBUFS Build SHOW QIO buffers'
: 1285      1273 1 GLOBAL ROUTINE NML$BLDSHOWBUFS (ENTITY, ENT_FORMAT, ENTITY_ADR,
: 1286      1274 1                                     NFB, P2_BUF_DSC, P2DSC,
: 1287      1275 1                                     QUAL_PST, QUAL_LEN, QUAL_ADR) =
: 1288      1276 1
: 1289      1277 1 ++
: 1290      1278 1 FUNCTIONAL DESCRIPTION:
: 1291      1279 1     This routine is called to finish the NFB buffer and build the P2
: 1292      1280 1     buffer for various special purpose SHOW operations. It is used
: 1293      1281 1     mostly when processing SHOW KNOWN or ACTIVE commands.
: 1294      1282 1
: 1295      1283 1 FORMAL PARAMETERS:
: 1296      1284 1
: 1297      1285 1     ENTITY      Entity type code.
: 1298      1286 1     ENT_FORMAT  NMA$C_ENT_KNO => Get KNOWN entities.
: 1299      1287 1                               NMA$C_ENT_ACT => Get ACTIVE entities.
: 1300      1288 1                               NMA$C_ENT_LOO => Get loop nodes.
: 1301      1289 1                               NMA$C_ENT_ADJ => Get adjacent nodes.
: 1302      1290 1     Length of entity ID (used for SHOW commands with
: 1303      1291 1     qualifiers. The qualifier makes the SHOW essentially
: 1304      1292 1     a multiple SHOW.
: 1305      1293 1     ENTITY_ADR  Address of entity ID string. Used only for SHOWs
: 1306      1294 1     with qualifiers.
: 1307      1295 1     NFB      Address of buffer with NFB to do single entity SHOW.
: 1308      1296 1     This buffer is modified to do SHOW KNOWN or ACTIVE.
: 1309      1297 1     P2_BUF_DSC  Address of descriptor of buffer in which to build
: 1310      1298 1     P2 info.
: 1311      1299 1     P2DSC      Address of descriptor of P2 info returned to caller.
: 1312      1300 1     QUAL_PST   Address of Qualifier's entry in the Parameter
: 1313      1301 1     Semantic Table (PST).
: 1314      1302 1     QUAL_LEN   Qualifier ID string length.
: 1315      1303 1     QUAL_ADR   Qualifier ID string address.
: 1316      1304 1
: 1317      1305 1 --
: 1318      1306 1
: 1319      1307 2 BEGIN
: 1320      1308 2
: 1321      1309 2 MAP
: 1322      1310 2     NFB:      REF BBLOCK,
: 1323      1311 2     QUAL_PST:  REF BBLOCK;
: 1324      1312 2
: 1325      1313 2 LOCAL
: 1326      1314 2     STATUS,
: 1327      1315 2     SEARCH_VAL1,
: 1328      1316 2     SEARCH_LEN1,
: 1329      1317 2     SEARCH_VAL2,
: 1330      1318 2     SEARCH_LEN2;
: 1331      1319 2
: 1332      1320 2
: 1333      1321 2 First fill in the NFB. This block describes the QIO to the ACP.
: 1334      1322 2
: 1335      1323 2 Set the MULTIPLE bit so the ACP returns multiple links in each buffer,
: 1336      1324 2 and the ERROR UPDATE bit, so the ACP will update it's pointer into it's
: 1337      1325 2 database even if an error is encountered in the search.
: 1338      1326 2
: 1339      1327 2 NFB [NFB$B_FLAGS] = NFB$M_MULT OR NFB$M_ERRUPD;
: 1340      1328 2 SELECTONEU'.ENT_FORMAT OF
```

```
1341 1329 2 SET
1342 1330 2
1343 1331 2 Set up the NFB to request SHOW KNOWN entities, SHOW ADJACENT NODES,
1344 1332 2 or SHOW LOOP NODES.
1345 1333 2
1346 1334 2 [NML$C_ENT_KNO, NML$C_ENT_LOO, NML$C_ENT_ADJ]:
1347 1335 2 BEGIN
1348 1336 2 NFB [NFB$S_SRCH_KEY] = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_KNO_SRCH_ID1];
1349 1337 2 NFB [NFB$B_OPER] = .NML$AB_ENTITYDATA [.ENTITY, EIT$B_KNO_OPER1];
1350 1338 2 SEARCH_VAL1 = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_KNO_SRCH_VAL1];
1351 1339 2 SEARCH_LEN1 = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_KNO_SRCH_LEN1];
1352 1340 2 END;
1353 1341 2
1354 1342 2 Set up the NFB to request SHOW ACTIVE entities.
1355 1343 2
1356 1344 2 [NML$C_ENT_ACT]:
1357 1345 2 BEGIN
1358 1346 2 NFB [NFB$S_SRCH_KEY] = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_ACT_SRCH_ID1];
1359 1347 2 NFB [NFB$B_OPER] = .NML$AB_ENTITYDATA [.ENTITY, EIT$B_ACT_OPER1];
1360 1348 2 SEARCH_VAL1 = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_ACT_SRCH_VAL1];
1361 1349 2 SEARCH_LEN1 = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_ACT_SRCH_LEN1];
1362 1350 2 END;
1363 1351 2
1364 1352 2 This path is useful for single entity SHOWs or SHOW commands with
1365 1353 2 qualifiers. For example, since the X25 GROUP qualifier, DTE, repeats
1366 1354 2 for a single GROUP, the SHOW command is essentially a multiple
1367 1355 2 operation.
1368 1356 2
1369 1357 2 [1 TO 16]:
1370 1358 2 BEGIN
1371 1359 2 NFB [NFB$S_SRCH_KEY] = .NML$AB_ENTITYDATA [.ENTITY, EIT$S_SRCH_ID1];
1372 1360 2 NFB [NFB$B_OPER] = NFB$C_OP_EQ;
1373 1361 2 SEARCH_VAL1 = .ENTITY_ADR;
1374 1362 2 SEARCH_LEN1 = .ENT_FORMAT;
1375 1363 2 END;
1376 1364 2 TES;
1377 1365 2
1378 1366 2
1379 1367 2 If there's a qualifier on the NICE command, use it for the second search
1380 1368 2 key. Otherwise, default the second search key to a wildcard.
1381 1369 2 Also, default the second search key to a wildcard if the entity id
1382 1370 2 is for circuits or nodes because the qualifiers for them are, respectively,
1383 1371 2 ADJACENT NODE and CIRCUIT, and are held in the adjacency database (AJI)
1384 1372 2 rather than the node or circuit databases.
1385 1373 2
1386 1374 2 NFB [NFB$B_OPER2] = NFB$C_OP_EQ;
1387 1375 2 IF .NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] AND
1388 1376 2 .ENTITY_NEQ_NML$C_CIRCUIT AND
1389 1377 2 .ENTITY_NEQ_NML$C_LOOPNODE AND
1390 1378 2 .ENTITY_NEQ_NML$C_ADJACENT_NODE THEN
1391 1379 2 BEGIN
1392 1380 2 NFB [NFB$S_SRCH2_KEY] = .QUAL_PST [PST$S_NFBID];
1393 1381 2 IF .QUAL_LEN EQ 0 THEN
1394 1382 2 SEARCH_VAL2 = ...QUAL_ADR
1395 1383 2 ELSE
1396 1384 2 SEARCH_VAL2 = ..QUAL_ADR;
1397 1385 2 SEARCH_LEN2 = .QUAL_LEN;
```



```
: 1398      1386 3      END
: 1399      1387 2      ELSE
: 1400      1388 3      BEGIN
: 1401      1389 3      SELECTONEU .ENTITY OF
: 1402      1390 3      SET
: 1403      1391 3      [NML$C_NODE]:
: 1404      1392 3      |
: 1405      1393 3      |   For multiple node shows, don't return the executor or loopnodes.
: 1406      1394 3      |   They are done separately. Note that using a second search key of
: 1407      1395 3      |   node address neq 0 filters out both the executor and loopnodes.
: 1408      1396 3      |   All loopnodes have an address of 0.
: 1409      1397 3      |
: 1410      1398 4      BEGIN
: 1411      1399 4      NFB [NFB$L_SRCH2_KEY] = NFB$C_NDI_ADD;
: 1412      1400 4      SEARCH_VAL2 = 0;
: 1413      1401 4      SEARCH_LEN2 = 0;
: 1414      1402 4      NFB [NFB$B_OPER2] = NFB$C_OP_NEQ;
: 1415      1403 3      END;
: 1416      1404 3      [NML$C_CIRCUIT_ADJACENT]:
: 1417      1405 3      |
: 1418      1406 3      |   For showing the ADJACENT NODES of SHOW CIRC, skip over entries
: 1419      1407 3      |   for which the node isn't reachable.
: 1420      1408 3      |
: 1421      1409 3      BEGIN
: 1422      1410 4      NFB [NFB$L_SRCH2_KEY] = NFB$C_AJI_REA;
: 1423      1411 4      SEARCH_VAL2 = 1;
: 1424      1412 4      SEARCH_LEN2 = 0;
: 1425      1413 4      END;
: 1426      1414 3      [OTHERWISE]:
: 1427      1415 3      BEGIN
: 1428      1416 4      NFB [NFB$L_SRCH2_KEY] = NFB$C_WILDCARD;
: 1429      1417 4      SEARCH_VAL2 = 0;
: 1430      1418 4      SEARCH_LEN2 = -1;
: 1431      1419 4      END;
: 1432      1420 3      TES;
: 1433      1421 3      END;
: 1434      1422 2      |
: 1435      1423 2      |   Build the P2 QIO buffer.
: 1436      1424 2      |
: 1437      1425 2      |   STATUS = NML$BLDP2 (.SEARCH_LEN1, .SEARCH_VAL1,
: 1438      1426 2      |   .SEARCH_LEN2, .SEARCH_VAL2,
: 1439      1427 2      |   .P2_BUF_DSC, .P2DSC);
: 1440      1428 2      RETURN .STATUS;
: 1441      1429 2      |
: 1442      1430 2      |
: 1443      1431 2      |
: 1444      1432 1      END;          ! End of NML$BLDSHOWBUFS
```

		003C 00000	.ENTRY	NML\$BLDSHOWBUFS, Save R2,R3,R4,R5	: 1273
	55	00000000G	00	9E 00002	
	51	10	AC	D0 00009	: 1327
01	A1		03	90 0000D	
	52	08	AC	D0 00011	: 1328

MOVAB NML\$AB_ENTITYDATA+14, R5
MOVL NFB, RT
MOVB #3, 1(R1)
MOVL ENT_FORMAT, R2

FFFFFFFC	8F	52	D1	00015	CMPL	R2, #-4	1334
		09	1F	0001C	BLSSU	1\$	
FFFFFFFD	8F	52	D1	0001E	CMPL	R2, #-3	
		09	1B	00025	BLEQU	2\$	
FFFFFFF	8F	52	D1	00027	CMPL	R2, #-1	
		1F	12	0002E	BNEQ	3\$	
50	04	AC	2C	C5	MULL3	#44, ENTITY, R0	1336
			6540	9F	PUSHAB	NML\$AB_ENTITYDATA+14[R0]	
	04	A1	9E	D0	MOVL	@(SP)+, 4(R1)	
	03	A1	0C	A540	MOVB	NML\$AB_ENTITYDATA+26[R0], 3(R1)	1337
			08	A540	PUSHAB	NML\$AB_ENTITYDATA+22[R0]	1338
		54	9E	D0	MOVL	@(SP)+, SEARCH_VAL1	
			04	A540	PUSHAB	NML\$AB_ENTITYDATA+18[R0]	1339
			27	11	BRB	4\$	
FFFFFFFE	8F	52	D1	0004F	CMPL	R2, #-2	1344
		23	12	00056	BNEQ	5\$	
50	04	AC	2C	C5	MULL3	#44, ENTITY, R0	1346
			0D	A540	PUSHAB	NML\$AB_ENTITYDATA+27[R0]	
	04	A1	9E	D0	MOVL	@(SP)+, 4(R1)	
	03	A1	19	A540	MOVB	NML\$AB_ENTITYDATA+39[R0], 3(R1)	1347
			15	A540	PUSHAB	NML\$AB_ENTITYDATA+35[R0]	1348
		54	9E	D0	MOVL	@(SP)+, SEARCH_VAL1	
			11	A540	PUSHAB	NML\$AB_ENTITYDATA+31[R0]	1349
		53	9E	D0	MOVL	@(SP)+, SEARCH_LEN1	
			20	11	BRB	6\$	1328
			52	D5	TSTL	R2	1357
			1C	13	BEQL	6\$	
	10		52	D1	CMPL	R2, #16	
			17	1A	BGTRU	6\$	
50	04	AC	2C	C5	MULL3	#44, ENTITY, R0	1359
			F8	A540	PUSHAB	NML\$AB_ENTITYDATA+6[R0]	
	04	A1	9E	D0	MOVL	@(SP)+, 4(R1)	
			03	A1	CLRB	3(R1)	1360
		54	0C	AC	MOVL	ENTITY_ADR, SEARCH_VAL1	1361
		53	52	D0	MOVL	R2, SEARCH_LEN1	1362
			0C	A1	CLRB	12(R1)	1374
33 00000000G	00	02	E1	0009E	BBC	#2, NML\$GL_PRS_FLGS, 9\$	1375
	09	04	AC	D1	CMPL	ENTITY, #9	1376
			2D	13	BEQL	9\$	
	05	04	AC	D1	CMPL	ENTITY, #5	1377
			27	13	BEQL	9\$	
	06	04	AC	D1	CMPL	ENTITY, #6	1378
			21	13	BEQL	9\$	
	50	1C	AC	D0	MOVL	QUAL PST, R0	1380
	08	A1	0C	A0	MOVL	12(R0), 8(R1)	
			20	AC	TSTL	QUAL_LEN	1381
			09	12	BNEQ	7\$	
	50	24	BC	D0	MOVL	@QUAL_ADR, R0	1382
	50		60	D0	MOVL	(R0), SEARCH_VAL2	
			04	11	BRB	8\$	
	50	24	BC	D0	MOVL	@QUAL_ADR, SEARCH_VAL2	1384
	52	20	AC	D0	MOVL	QUAL_LEN, SEARCH_LEN2	1385
			38	11	BRB	12\$	1375
	50	04	AC	D0	MOVL	ENTITY, R0	1389
	03		50	D1	CMPL	R0, #3	1391
			12	12	BNEQ	10\$	
08	A1	02010012	8F	D0	MOVL	#33619986, 8(R1)	1399

NML\$SHOW
V04-000

NML SHOW parameter module
NML\$BLDSHOWBUFS Build SHOW QIO buffers

D 8
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NMLSHOW.B32;1 Page 53
(13)

		50	D4	000EA	CLRL	SEARCH_VAL2	:	1400
		52	D4	000EC	CLRL	SEARCH_LEN2	:	1401
0C	A1	03	90	000EE	MOVB	#3, 12(R1)	:	1402
		1D	11	000F2	BRB	12\$:	1389
	0A	50	D1	000F4	CMPL	R0, #10	:	1405
		0F	12	000F7	BNEQ	11\$:	
08	A1	8F	D0	000F9	MOVL	#318767106, 8(R1)	:	1411
	50	01	D0	00101	MOVL	#1, SEARCH_VAL2	:	1412
		52	D4	00104	CLRL	SEARCH_LEN2	:	1413
		09	11	00106	BRB	12\$:	1389
08	A1	01	D0	00108	MOVL	#1, 8(R1)	:	1418
		50	D4	0010C	CLRL	SEARCH_VAL2	:	1419
	52	01	CE	0010E	MNEGL	#1, SEARCH_LEN2	:	1420
	7E	14	AC	7D 00111	MOVQ	P2_BUF_DSC, -(SP)	:	1429
		50	DD	00115	PUSHL	SEARCH_VAL2	:	1428
		52	DD	00117	PUSHL	SEARCH_LEN2	:	
		18	BB	00119	PUSHR	#*M<R3,R4>	:	1427
00000000G	00	06	FB	0011B	CALLS	#6, NML\$BLDP2	:	
		04	00	122	RET		:	1432

; Routine Size: 291 bytes, Routine Base: \$CODE\$ + 0870


```
: 1446 1433 1 %SBTTL 'NML$GETDATA Get volatile entity data'
: 1447 1434 1 GLOBAL ROUTINE NML$GETDATA (NFB DSC, P2 DSC, QBFDSC, P4_DATA_DSC) =
: 1448 1435 1
: 1449 1436 1 !++
: 1450 1437 1 FUNCTIONAL DESCRIPTION:
: 1451 1438 1
: 1452 1439 1 This routine reads volatile entity data for the specified NFB and
: 1453 1440 1 P2 parameters.
: 1454 1441 1
: 1455 1442 1 FORMAL PARAMETERS:
: 1456 1443 1
: 1457 1444 1 NFB DSC Address of NFB descriptor.
: 1458 1445 1 P2 DSC Address of P2 descriptor.
: 1459 1446 1 QBFDSC Address of QIO buffer descriptor.
: 1460 1447 1 P4_DATA_DSC Address of descriptor for data to be read.
: 1461 1448 1
: 1462 1449 1 !--
: 1463 1450 1
: 1464 1451 2 BEGIN
: 1465 1452 2
: 1466 1453 2 MAP
: 1467 1454 2 NFB DSC : REF DESCRIPTOR,
: 1468 1455 2 P2 DSC : REF DESCRIPTOR,
: 1469 1456 2 QBFDSC : REF DESCRIPTOR,
: 1470 1457 2 P4_DATA_DSC : REF DESCRIPTOR;
: 1471 1458 2
: 1472 1459 2 LOCAL
: 1473 1460 2 STATUS;
: 1474 1461 2
: 1475 1462 2 IF .QBFDSC NEQ 0 THEN
: 1476 1463 2 P4_DATA_DSC [DSC$A_POINTER] = .QBFDSC [DSC$A_POINTER];
: 1477 1464 2
: 1478 1465 2 STATUS = NML$NETQIO (.NFB DSC,
: 1479 1466 2 P2 DSC,
: 1480 1467 2 P4_DATA_DSC [DSC$W_LENGTH],
: 1481 1468 2 .QBFDSC);
: 1482 1469 2
: 1483 1470 2 RETURN .STATUS
: 1484 1471 2
: 1485 1472 1 END; ! End of NML$GETDATA
```

			0000	C0000	.ENTRY	NML\$GETDATA, Save nothing	: 1434
	51	0C	AC	D0 00002	MOVL	QBFDSC, R1	: 1462
			09	13 00006	BEQL	1\$:
	50	10	AC	D0 00008	MOVL	P4_DATA_DSC, R0	: 1463
04	A0	04	A1	D0 0000C	MOVL	4(R1), 4(R0)	:
			51	DD 00011	PUSHL	R1	: 1468
		10	AC	DD 00013	PUSHL	P4_DATA_DSC	: 1467
	7E	04	AC	7D 00016	MOVQ	NFB DSC, --(SP)	:
00000000G	00		04	FB 0001A	CALLS	#4, NML\$NETQIO	:
			04	00021	RET		: 1472

; Routine Size: 34 bytes, Routine Base: \$CODE\$ + 0993

NML\$SHOW
V04-000

NML SHOW parameter module
NML\$GETDATA Get volatile entity data

F 8
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1 Page 55
(14)

NML
VO


```
1487 1 %SBTTL 'NML$PROCESSDATA Add data to output message'
1488 1 GLOBAL ROUTINE NML$PROCESSDATA (ENT, TABDES, P4_DATA_DSC,
1489 1 P4_DATA_PTR, NICE_MSG_DSC) :NOVALUE =
1490 1
1491 1 !++
1492 1 FUNCTIONAL DESCRIPTION:
1493 1
1494 1 This routine adds data to the output message using the information
1495 1 table and the input data buffer.
1496 1
1497 1 FORMAL PARAMETERS:
1498 1
1499 1 ENT Internal entity id code.
1500 1 TABDES Address of information table descriptor.
1501 1 P4_DATA_DSC Address of data buffer descriptor.
1502 1 P4_DATA_PTR Address of data buffer pointer.
1503 1 NICE_MSG_DSC Address of descriptor to describe output message.
1504 1
1505 1 !--
1506 1
1507 2 BEGIN
1508 2
1509 2 MAP
1510 2 tabdes : REF DESCRIPTOR,
1511 2 p4_data_dsc : REF DESCRIPTOR,
1512 2 nice_msg_dsc : REF DESCRIPTOR;
1513 2
1514 2 LOCAL
1515 2 msgsize, ! Output message length
1516 2 strdsc : DESCRIPTOR; ! Entity id string descriptor
1517 2
1518 2 nml$getidstring (.ent, .p4_data_ptr, strdsc); ! Get entity id
1519 2 nml$ab_msgblock [msb$_flags] = msb$m_entd_fld;
1520 2 nml$ab_msgblock [msb$b_code] = nma$c_sts_suc;
1521 2 nml$ab_msgblock [msb$a_entity] = strdsc;
1522 2
1523 2 nml$bld_reply (nml$ab_msgblock, msgsize);
1524 2
1525 2 nml$showparlist (nml$gq_sndbfdsc,
1526 2 msgsize,
1527 2 .tabdes,
1528 2 .p4_data_dsc,
1529 2 .p4_data_ptr);
1530 2
1531 2 nice_msg_dsc [dsc$w_length] = .msgsize;
1532 2 nice_msg_dsc [dsc$a_pointer] = .nml$gq_sndbfdsc [dsc$a_pointer];
1533 1 END; ! End of NML$PROCESSDATA
```

```
0004 00000
52 00000000G 00 9E 00002
5E 0C C2 00009
04 AE 9F 0000C
10 AC DD 0000F
```

```
.ENTRY NML$PROCESSDATA, Save R2
MOVAB NML$AB_MSGBLOCK, R2
SUBL2 #12, SP
PUSHAB STRDSC
PUSHL P4_DATA_PTR
```

```
: 1474
:
:
: 1504
:
```


NML\$SHOW
V04-000

NML SHOW parameter module
NML\$PROCESSDATA Add data to output message

H 8
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1 Page 57
(15)

00000000V	00	04	AC	DD	00012	PUSHL	ENT	:
	62		03	FB	00015	CALLS	#3, NML\$GETIDSTRING	:
04	A2		10	DO	0001C	MOVL	#16, NML\$AB_MSGBLOCK	1505
14	A2		01	90	0001F	MOVB	#1, NML\$AB_MSGBLOCK+4	1506
		04	AE	9E	00023	MOVAB	STRDSC, NML\$AB_MSGBLOCK+20	1507
		4004	8F	BB	00028	PUSHR	#^M<R2,SP>	1509
00000000G	00		02	FB	0002C	CALLS	#2, NML\$BLD_REPLY	:
	7E		AC	7D	00033	MOVQ	P4 DATA_DSC, -(SP)	1514
		0C	AC	DD	00037	PUSHL	TABDES	1513
		08	AC	DD	00037	PUSHAB	MSGSIZE	1511
		0C	AE	9F	0003A	PUSHAB	NML\$GQ_SNDBFDSC	:
		00000000G	00	9F	0003D	CALLS	#5, NML\$SHOWPARLIST	:
00000000G	00		05	FB	00043	MOVL	NICE_MSG_DSC, R0	1517
	50	14	AC	DO	0004A	MOVW	MSGSIZE, -(R0)	:
	60		6E	BO	0004E	MOVL	NML\$GQ_SNDBFDSC+4, 4(R0)	1518
04	A0	00000000G	00	DO	00051	RET		1519
			04	00	00059			:

; Routine Size: 90 bytes, Routine Base: \$CODE\$ + 09B5

NML
V04


```
: 1535 1520 1 %SBTTL 'NML$GETIDSTRING Get entity id string'
: 1536 1521 1 GLOBAL ROUTINE NML$GETIDSTRING (ENT, P4_DATA_PTR, STRDSC) =
: 1537 1522 1
: 1538 1523 1 !++
: 1539 1524 1 FUNCTIONAL DESCRIPTION:
: 1540 1525 1
: 1541 1526 1 This routine builds the entity id string and descriptor for the
: 1542 1527 1 NICE response message. It gets the entity ID from the P4 buffer
: 1543 1528 1 returned by NETACP.
: 1544 1529 1
: 1545 1530 1 FORMAL PARAMETERS:
: 1546 1531 1
: 1547 1532 1 ENT Internal entity id code.
: 1548 1533 1 P4_DATA_PTR Address of data buffer pointer.
: 1549 1534 1 STRDSC Address of descriptor for output id string.
: 1550 1535 1
: 1551 1536 1 !--
: 1552 1537 1
: 1553 1538 2 BEGIN
: 1554 1539 2
: 1555 1540 2 MAP
: 1556 1541 2 STRDSC : REF DESCRIPTOR;
: 1557 1542 2
: 1558 1543 2 LOCAL
: 1559 1544 2 LEN,
: 1560 1545 2 PTR;
: 1561 1546 2
: 1562 1547 2 STRDSC [DSC$A_POINTER] = .NML$Q_ENTBFDSC [DSC$A_POINTER];
: 1563 1548 2 PTR = .STRDSC [DSC$A_POINTER];
: 1564 1549 2
: 1565 1550 2 SELECTONEU .ENT OF
: 1566 1551 2 SET
: 1567 1552 2
: 1568 1553 2 [NML$C_CIRCUIT,
: 1569 1554 2 NML$C_CIRCUIT_ADJACENT,
: 1570 1555 2 NML$C_CIRCUIT_ADJ_SRV,
: 1571 1556 2 NML$C_LINE,
: 1572 1557 2 NML$C_OBJECT]:
: 1573 1558 2 BEGIN
: 1574 1559 2
: 1575 1560 2 LEN = CH$RCHAR_A (.P4_DATA_PTR);
: 1576 1561 2 CH$RCHAR_A (.P4_DATA_PTR);
: 1577 1562 2
: 1578 1563 2 CH$WCHAR_A (.LEN, PTR);
: 1579 1564 2 PTR = CH$MOVE (.LEN, ..P4_DATA_PTR, .PTR);
: 1580 1565 2
: 1581 1566 2 .P4_DATA_PTR = ..P4_DATA_PTR + .LEN;
: 1582 1567 2
: 1583 1568 2 END;
: 1584 1569 2
: 1585 1570 2 [NML$C_LOGGING, NML$C_SINK]:
: 1586 1571 2 ;
: 1587 1572 2
: 1588 1573 2 [NML$C_LOOPNODE]:
: 1589 1574 2 BEGIN
: 1590 1575 2
: 1591 1576 2 .P4_DATA_PTR = ..P4_DATA_PTR + 4; ! Skip address (always 0)
```

```
1592 1577 3
1593 1578 CH$WCHAR_A (0, PTR); ! Move 0 address
1594 1579 CH$WCHAR_A (0, PTR);
1595 1580
1596 1581 LEN = (...P4_DATA_PTR)<0,16>; ! Move name
1597 1582 .P4_DATA_PTR = ..P4_DATA_PTR + 2;
1598 1583 CH$WCHAR_A (.LEN, PTR);
1599 1584 PTR = CH$MOVE (.LEN, ..P4_DATA_PTR, .PTR);
1600 1585
1601 1586 .P4_DATA_PTR = ..P4_DATA_PTR + .LEN;
1602 1587
1603 1588 END;
1604 1589
1605 1590 [NML$C_LINKS]:
1606 1591 BEGIN
1607 1592 CH$WCHAR_A (0, PTR);
1608 1593 PTR = CH$MOVE (2, ..P4_DATA_PTR, .PTR); ! Move link number.
1609 1594 .P4_DATA_PTR = ..P4_DATA_PTR + 4;
1610 1595 END;
1611 1596
1612 1597 [NML$C_X25_ACCESS]:
1613 1598 $MOVE_ASCII ('X25-ACCESS', PTR);
1614 1599
1615 1600 [NML$C_PROT_NET,
1616 1601 NML$C_PROT_DTE,
1617 1602 NML$C_PROT_GRP]:
1618 1603 $MOVE_ASCII ('X25-PROTOCOL', PTR);
1619 1604
1620 1605 [NML$C_X25_SERV,
1621 1606 NML$C_X25_SERV_DEST]:
1622 1607 $MOVE_ASCII ('X25-SERVER', PTR);
1623 1608
1624 1609 [NML$C_TRACE,
1625 1610 NML$C_TRACE_PNT]:
1626 1611 $MOVE_ASCII ('X25-TRACE', PTR);
1627 1612
1628 1613 [NML$C_X29_SERV,
1629 1614 NML$C_X29_SERV_DEST]:
1630 1615 $MOVE_ASCII ('X29-SERVER', PTR);
1631 1616
1632 1617 [NML$C_AREA]:
1633 1618 BEGIN
1634 1619 CH$WCHAR_A (0, PTR); ! 0 means area address
1635 1620 ! follows.
1636 1621 CH$WCHAR_A (...P4_DATA_PTR, PTR); ! Move area address.
1637 1622 .P4_DATA_PTR = ..P4_DATA_PTR + 4; ! Increment P4 buffer pointer.
1638 1623 END;
1639 1624
1640 1625 [OTHERWISE]: ! It's a remote node or the executor.
1641 1626 BEGIN
1642 1627
1643 1628 ! If I'm talking to a Phase III NCP, and the entity is a node
1644 1629 ! outside the executor's area, don't return the node to the NCP.
1645 1630 ! Phase III doesn't include areas. If it's a Phase III NCP and
1646 1631 ! the node is in the executor's area, clear the area number from
1647 1632 ! the node number.
1648 1633
```



```
: 1649      1634 3      IF CH$RCHAR (nml$gb_ncp_version) LEQ 3 THEN
: 1650      1635 4      BEGIN
: 1651      1636 4      BIND node_addr = ..p4_data_ptr : BBLOCK;
: 1652      1637 4
: 1653      1638 4      IF .node_addr [nma$u_area] EQL
: 1654      1639 4      .nml$gw_vol_exec_addr [nma$u_area] THEN
: 1655      1640 4      node_addr [nma$u_area] = 0;
: 1656      1641 3      END;
: 1657      1642 3      ptr = CH$MOVE (2, ..p4_data_ptr, .ptr); ! Move address
: 1658      1643 3      .p4_data_ptr = ..p4_data_ptr + 4;
: 1659      1644 3
: 1660      1645 3      len = ..p4_data_ptr < 0, 16 >; ! Move name
: 1661      1646 3      .p4_data_ptr = ..p4_data_ptr + 2;
: 1662      1647 3      IF .ent EQL nml$sc_executor THEN
: 1663      1648 3      CH$WCHAR_A (.len OR nma$m_ent_exe, ptr)
: 1664      1649 3      ELSE
: 1665      1650 3      CH$WCHAR_A (.len, ptr);
: 1666      1651 3      ptr = CH$MOVE (.len, ..p4_data_ptr, .ptr);
: 1667      1652 3
: 1668      1653 3      .p4_data_ptr = ..p4_data_ptr + .len;
: 1669      1654 3
: 1670      1655 2      END;
: 1671      1656 2
: 1672      1657 2      TES;
: 1673      1658 2
: 1674      1659 2      strdsc [dsc$w_length] = .ptr - .strdsc [dsc$a_pointer];
: 1675      1660 2      RETURN nml$sts_suc;
: 1676      1661 1      END; ! End of NML$GETIDSTRING
```

```
4C 4F 53 53 45 43 43 41 2D 35 32 58 0A 000A0 P.AAU: .ASCII <10>\X25-ACCESS\
43 4F 54 4F 52 50 2D 35 32 58 0C 000AB P.AAV: .ASCII <12>\X25-PROTOCOL\
52 45 56 52 45 53 2D 35 32 58 0A 000B8 P.AAW: .ASCII <10>\X25-SERVER\
45 43 41 52 54 2D 35 32 58 09 000C3 P.AAX: .ASCII <9>\X25-TRACE\
52 45 56 52 45 53 2D 39 32 58 0A 000CD P.AAY: .ASCII <10>\X29-SERVER\
```

```
.PSECT $CODE$,NOWRT,2
07FC 00000
.ENTRY NML$GETIDSTRING, Save R2,R3,R4,R5,R6,R7,R8,-; 1521
5A 00000000' 00 9E 00002 MOVAB NML$Q_ENTBFDSC+4, R10
58 0C AC D0 00009 MOVL STRDSC, R8
04 A8 6A D0 0000D MOVL NML$Q_ENTBFDSC+4, 4(R8)
53 04 A8 D0 00011 MOVL 4(R8), PTR
57 04 AC D0 00015 MOVL ENT, R7
08 57 D1 0001B BEQL 1$
23 1F 0001E CMPL R7, #8
0B 57 D1 00020 BLSSU 2$
1E 1A 00023 CMPL R7, #11
50 08 BC D0 00025 1$: BGTRU 2$
59 60 9A 00029 MOVL @P4_DATA_PTR, R0
MOVZBL (R0), LEN 1560
```

			08	BC	D6	0002C	INCL	@P4_DATA_PTR		
			08	BC	D6	0002F	INCL	@P4_DATA_PTR	1561	
		83		59	90	00032	MOVB	LEN, (PTR)+	1563	
63		50		08	BC	D0	00035	MOVL	@P4_DATA_PTR, R0	1564
		60		59	28	00039	MOVCL	LEN, (R0), (PTR)		
	08	BC		59	C0	0003D	ADDL2	LEN, @P4_DATA_PTR	1566	
				73	11	00041	BRB	9\$	1550	
				57	D5	00043	2\$: TSTL	R7	1570	
				05	13	00045	BEQL	3\$		
		02		57	D1	00047	CMPL	R7, #2		
				7C	1B	0004A	BLEQU	11\$		
		05		57	D1	0004C	3\$: CMPL	R7, #5	1573	
				13	12	0004F	BNEQ	4\$		
		56		08	AC	D0	00051	MOVL	P4_DATA_PTR, R6	1576
		66			04	C0	00055	ADDL2	#4, (R6)	
				83	B4	00058	CLRW	(PTR)+	1578	
		59		00	B6	3C	0005A	MOVZWL	@0(R6), LEN	1581
		66			02	C0	0005E	ADDL2	#2, (R6)	1582
				30BA	31	00061	BRW	17\$	1583	
		18			57	D1	00064	4\$: CMPL	R7, #24	1590
					0C	12	00067	BNEQ	5\$	
				83	94	00069	CLRB	(PTR)+	1592	
		50		08	AC	D0	0006B	MOVL	P4_DATA_PTR, R0	1593
		83		00	B0	B0	0006F	MOVW	@0(R0), -(PTR)+	
				64	11	00073	BRB	13\$	1594	
		0D			57	D1	00075	5\$: CMPL	R7, #13	1597
				08	12	00078	BNEQ	6\$		
63	008C	CA		0B	28	0007A	MOVCL	#11, P.AAU, (PTR)	1598	
				5A	11	00080	BRB	14\$		
		0E			57	D1	00082	6\$: CMPL	R7, #14	1600
				0D	1F	00085	BLSSU	7\$		
		10			57	D1	00087	CMPL	R7, #16	
				08	1A	0008A	BGTRU	7\$		
63	0097	CA		0D	28	0008C	MOVCL	#13, P.AAV, (PTR)	1603	
				48	11	00092	BRB	14\$		
		11			57	D1	00094	7\$: CMPL	R7, #17	1605
				0D	1F	00097	BLSSU	8\$		
		12			57	D1	00099	CMPL	R7, #18	
				08	1A	0009C	BGTRU	8\$		
63	00A4	CA		0B	28	0009E	MOVCL	#11, P.AAW, (PTR)	1607	
				36	11	000A4	BRB	14\$		
		13			57	D1	000A6	8\$: CMPL	R7, #19	1609
				0D	1F	000A9	BLSSU	10\$		
		14			57	D1	000AB	CMPL	R7, #20	
				08	1A	000AE	BGTRU	10\$		
63	00AF	CA		0A	28	000B0	MOVCL	#10, P.AAX, (PTR)	1611	
				73	11	000B6	BRB	19\$		
		15			57	D1	000B8	9\$: CMPL	R7, #21	1613
				0D	1F	000BB	BLSSU	12\$		
		16			57	D1	000BD	CMPL	R7, #22	
				08	1A	000C0	BGTRU	12\$		
63	00B9	CA		0B	28	000C2	MOVCL	#11, P.AAY, (PTR)	1615	
				61	11	000C8	BRB	19\$		
		0C			57	D1	000CA	11\$: CMPL	R7, #12	1617
					0F	12	000CD	12\$: BNEQ	15\$	
				83	94	000CF	CLRB	(PTR)+	1619	
		50		08	AC	D0	000D1	MOVL	P4_DATA_PTR, R0	1621

NML\$SHOW
V04-000

NML SHOW parameter module
NML\$GETIDSTRING Get entity id string

M 8
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NML\$SHOW.B32;1 Page 62
(16)

		83	00	B0	90	000D5	MOVB	@0(R0), (PTR)+	:				
		60		04	C0	000D9	ADDL2	#4, (R0)	:	1622			
				4D	11	000DC	BRB	19\$:	1550			
		03	00000000G	00	91	000DE	15\$:	CMPB	NML\$GB_NCP_VERSION, #3	:	1634		
				19	1A	000E5	BGTRU	16\$:				
		50		08	BC	D0	000E7	MOVL	@P4_DATA_PTR, R0	:	1636		
51	00000000G	00		02	EF	000EB	EXTZV	#2, #6, NML\$GW_VOL_EXEC_ADDR+1, R1	:	1639			
51		60		0A	ED	000F4	CMPZV	#10, #6, (R0), -R1	:				
				05	12	000F9	BNEQ	16\$:				
		01	A0	FC	8F	8A	000FB	BICB2	#252, 1(R0)	:	1640		
			56	08	AC	D0	00100	16\$:	MOVL	P4_DATA_PTR, R6	:	1642	
			83	00	B6	B0	00104	MOVW	@0(R6), (PTR)+	:			
			66		04	C0	00108	ADDL2	#4, (R6)	:	1643		
			59	00	B6	3C	0010B	MOVZWL	@0(R6), LEN	:	1645		
			66		02	C0	0010F	ADDL2	#2, (R6)	:	1646		
			07		57	D1	00112	CMPL	R7, #7	:	1647		
					07	12	00115	BNEQ	17\$:			
		63	59	80	8F	89	00117	BISB3	#128, LEN, (PTR)	:	1648		
					03	11	0011C	BRB	18\$:	1650		
			63		59	90	0011E	17\$:	MOVB	LEN, (PTR)	:		
					53	D6	00121	18\$:	INCL	PTR	:	1648	
		63	00	B6	59	28	00123	MOVW3	LEN, @0(R6), (PTR)	:	1651		
				66	59	C0	00128	ADDL2	LEN, (R6)	:	1653		
		68		53	04	A8	A3	0012B	19\$:	SUBW3	4(R8), PTR, (R8)	:	1659
				50	01	D0	00130	MOVL	#1, R0	:	1660		
					04	00133	RET		:	1661			

; Routine Size: 308 bytes, Routine Base: \$CODE\$ + 0A0F

NML
V04

54

NML\$SHOW
V04-000

NML SHOW parameter module
NML\$GETIDSTRING Get entity id string

N 8
16-Sep-1984 00:34:50
14-Sep-1984 12:50:20

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[NML.SRC]NMLSHOW.B32;1 Page 63 (17)

: 1678
: 1679
: 1680

1662 1 END
1663 1
1664 0 ELUDOM

! End of module

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	2628	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	216	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	2883	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[NML.OBJ]NMLLIB.L32;1	341	73	21	27	00:00.1
_\$255\$DUA28:[SHRLIB]NMLIBRY.L32;1	887	8	0	47	00:00.2
_\$255\$DUA28:[SHRLIB]NET.L32;1	1279	43	3	63	00:00.3
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	2	0	581	00:03.3

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:NMLSHOW/OBJ=OBJ\$:NMLSHOW MSRC\$:NMLSHOW/UPDATE=(ENH\$:NMLSHOW)

: Size: 2883 code + 2844 data bytes
: Run Time: 00:55.6
: Elapsed Time: 02:16.2
: Lines/CPU Min: 1796
: Lexemes/CPU-Min: 20482
: Memory Used: 217 pages
: Compilation Complete

0287 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

